CHAPTER 10

Graphics Generation and Manipulation

TEX probably has the best algorithm for formatting paragraphs and building pages from them. But in this era of ever-increasing information exchange, most publications do not limit themselves to text—the importance of graphical material has grown tremendously. TEX by itself does not address this issue, as it deals only with positioning (black) boxes on a page. Knuth, however, provided a hook for implementing "features" that are not available in the basic language, via the `\special` command. The latter command does not affect the output page being formatted, but TEX will put the material, specified as an argument in the `\special` command, literally at the current point in the `.dvi` file.[1] The `dvi` driver then has to interpret the received information and produce the output image accordingly (see also [144]).

The *LaTeX Graphics Companion* [57, Chapter 1] describes in detail various approaches that can be used to produce graphics with TEX. The following list gives a short overview. Interested readers are referred to that book for more details.

1. ASCII drawing, such as PICTEX, which provides a complete plotting language where most graphical elements are implemented by combining a very large number of small dots.
2. Picture-element fonts, such as LaTeX's `picture` environment. Kristoffer Rose's xypic system [57, Chapter 5] uses special fonts to typeset diagrams.

[1] In certain situations the `\special` command may change the formatting because it can produce an additional breakpoint and it might prevent LaTeX from noticing spaces.

3. Picture macro packages, mainly based on the `picture` environment or on TeX's raw line-drawing commands. Among others, packages exist for drawing chemical formulae [57, Section 6.2], trees, and bar charts (see Section 10.1.6).

4. Picture fonts, where each character to be typeset is one, possibly enormous, "letter" in a font. One can use METAFONT or MetaPost for generating the pictures [57, Chapter3], or else use already existing bitmaps and transform them into a `.pk` file directly [57, Section 1.3].

5. Half-tone fonts—blocks consisting of various levels of grey, which can be combined in the normal TeX way to generate pictures [39, 93].

6. Graphics material included via the `\special` command. This approach is by definition device dependent, as it relies on the possibilities of the `dvi` driver and the output device. The graphics package, described in Section 10.2, offers a higher-level support layer on top of TeX's `\special` command. This approach has become very common because of the wide availability of low-cost PostScript printers and previewers. Other high-level systems allowing one to use PostScript together with LaTeX are psfrag and pstricks [57, Chapter4].

In this chapter we look at techniques for producing portable graphics (mainly based on item 3) and at the high-level interface to device-dependent graphics support (item 6).

In particular, the first section discusses LaTeX's built-in graphics tools. We look at how to build ornaments, which can be useful for making important material stand out. Then we turn our attention to two packages, epic and eepic, that extend the `picture` environment by introducing a set of new commands. They are described in detail and examples show how they are used in practice.

LaTeX 2_ε provides a generalized driver-independent interface to include external graphic material and to scale and rotate LaTeX boxes.[1] Section 10.2 deals with graphics file inclusion. For this LaTeX offers both a simple interface (graphics; see Section 10.2.2), which can be combined with the separate rotation and scaling commands, and a more complex interface (graphicx; see Section 10.2.3), which has its own powerful set of image manipulation options. Free-standing scaling and rotation is the subject of Section 10.3.

In the final section we say a few words about important display languages (PostScript, PDF, SVG). We also briefly discuss dvips, an often-used `dvi` to PostScript translation program, and describe pspicture, an extension of LaTeX's `picture` environment that uses PostScript drawing primitives interfaced to the dvips driver.

[1] A generalized package for color is also available; see the *LaTeX Manual* [104] for more details.

10.1 Producing portable graphics and ornaments

Portable graphics in LaTeX essentially mean graphics built from boxes, lines, and characters. LaTeX boxes are reviewed briefly in Appendix A.2. Here, we first present packages that provide extensions to the usual LaTeX boxes. Later, this section deals with line graphics.

10.1.1 boxedminipage—Boxes with frames

The `boxedminipage` environment, defined in the **boxedminipage** package (by Mario Wolczko), behaves like the standard `minipage` environment, but the result is surrounded by a frame, as if it was placed inside an `\fbox`. The thickness and separation of the rules are controlled by the `\fboxrule` and `\fboxsep` parameters, respectively. However, in contrast to a construction involving `\fbox`, one can use verbatim commands inside the environment body.

This is an example of a small boxed minipage sporting a footnote[a] and a `\verb` command.

[a]Very simple example

10-1-1

```
\usepackage{boxedminipage}
\begin{boxedminipage}[t]{5cm}
 This is an example of a small boxed  minipage
 sporting a footnote\footnote{Very simple example}
 and a \verb=\verb= command.
\end{boxedminipage}
```

10.1.2 shadow—Boxes with shadows

The **shadow** package (by Mauro Orlandini) defines the `\shabox` command. It is similar to the LaTeX command `\fbox`, except that a "shadow" is added to the bottom and the right side of the box.

Three parameters control the visual appearance of the box (defaults are given in parentheses): `\sboxrule` defines the width of the lines for the frame (0.4pt); `\sboxsep` defines the separation between the frame and the text (10pt); and `\sdim` specifies the dimension of the shadow (4pt).

A complete paragraph can be highlighted by putting it in a parbox, nested inside a `shabox`.

10-1-2

```
\usepackage{shadow}
\setlength\sdim{10pt}
\shabox{\parbox{6cm}{A complete
  paragraph can be highlighted by
  putting it in a parbox,
  nested inside a \texttt{shabox}.}}
```

10.1.3 fancybox—Ornamental boxes

Timothy Van Zandt, in the framework of his seminar package for producing slides, developed the fancybox package. It introduces various new commands for boxing and framing data in LaTeX. In this section we review only a few of the more basic commands. More information can be found in the documentation accompanying the seminar package.

The package introduces four variants for the `\fbox` command. As with the `\fbox` command, the distance between the box and the frame is given by the length parameter `\fboxsep` (LaTeX's default is 3pt). Other parameters governing these boxes are described below.

The `\shadowbox` command adds a shadow with width `\shadowsize` (default 4pt). The box is aligned at the base of the shadow, which makes it probably less suitable for inline usage than the `\shabox` command described earlier. Notice the different spacing defaults.

X This is a shadowbox Y This is a shabox Z

```
\usepackage{fancybox}
\usepackage{shadow}
X \shadowbox{This is a shadowbox}
Y \shabox{This is a shabox} Z
```

10-1-3

The `\ovalbox` command generates a frame with rounded corners. The width of the frame is the same as that produced by standard picture elements when the `\thinlines` declaration is in effect. The `\Ovalbox` command is similar but has a frame width corresponding to the size produced by a `\thicklines` declaration. The diameter of the corner arcs is set with a `\cornersize` declaration. The form `\cornersize{`*num*`}` sets the diameter to *num* $\times$ minimum (width of box, height of box); the form `\cornersize*{`*len*`}` sets the diameter to the length *len*. The default is `\cornersize{0.5}`.

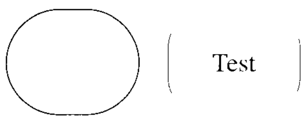
This is an ovalbox This is an ovalbox

This is an
Ovalbox

```
\usepackage{fancybox}
\centering
            \ovalbox{This is an ovalbox}
\cornersize{1} \ovalbox{This is an ovalbox}
\\[8pt]
\setlength\fboxsep{6pt} \cornersize*{7mm}
\Ovalbox{\shortstack{This is an\\Ovalbox}}
```

10-1-4

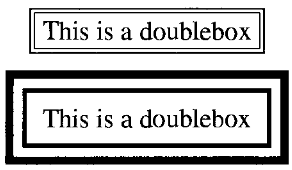
The package also provides `\fancyoval` as an alternative to LaTeX's `\oval` picture command. While `\oval` always makes the diameter of the corner arcs as large as possible, `\fancyoval` uses the `\cornersize` declaration to set the diameter.




```
\usepackage{fancybox,color}
\cornersize{0.7}
\begin{picture}(110,40)
  \put(25,20){\oval(50,40)}
\color{blue}
  \put(85,20){\makebox(0,0){Test}}
  \put(85,20){\fancyoval(50,40)}
\end{picture}
```

10-1-5

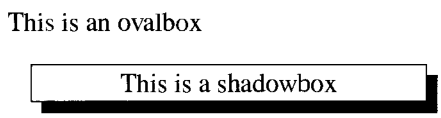
Finally, the package offers the \doublebox command, which generates two square frames. Their widths and relations to each other and the text are fractions of the \fboxrule parameter value: the width of the inner frame is 0.75 of \fboxrule and that of the outer frame is 1.5 of \fboxrule. The distance between the two frames is 1.5 of \fboxrule plus 0.5pt.




```
\usepackage{fancybox}
\centering
\doublebox{This is a doublebox} \\[5pt]
\setlength\fboxsep {6pt}    % default 3pt
\setlength\fboxrule{2pt}
\doublebox{This is a doublebox}
```

10-1-6

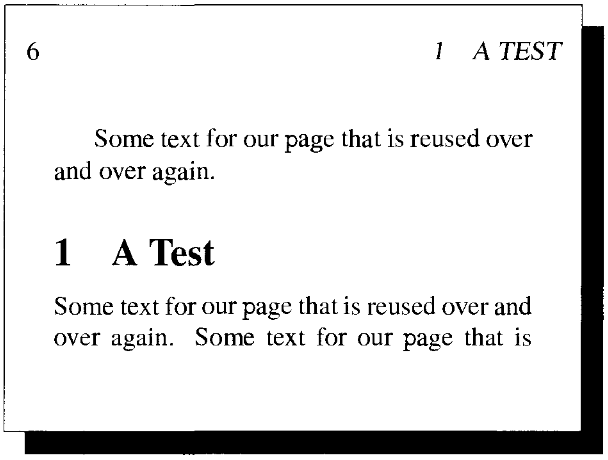
None of the above commands have optional arguments, unlike \framebox and \makebox. You can get exactly the same functionality by using \makebox in the argument of these framing commands.




```
\usepackage{fancybox}
\centering
\cornersize{0.8}
\ovalbox{\makebox[6cm][l]
         {This is an ovalbox}}  \\[8pt]
\shadowbox{\makebox[5cm]
         {This is a shadowbox}}
```

10-1-7

For some types of documents, such as slides, it would be nice to allow for framed pages—that is, to apply commands like those introduced in this section as part of the page style. This capability is supported by the fancybox package through the declaration \fancypage{*inner*}{*outer*}. The completed page, before headers and footers are added, is boxed (so it has width \textwidth and height \textheight) and then passed to the code specified in *inner* as an argument. Next the headers and footers are added using the new width of the page, in case it is changed by *inner*. The result is passed as an argument to the code in *outer*, which again expects one argument. Thus, in the simplest case, you could specify one of the boxing commands from this section, or even leave one of the arguments empty. The next example shows an application where the arguments also contain some parameter settings to influence the form of the added frames.

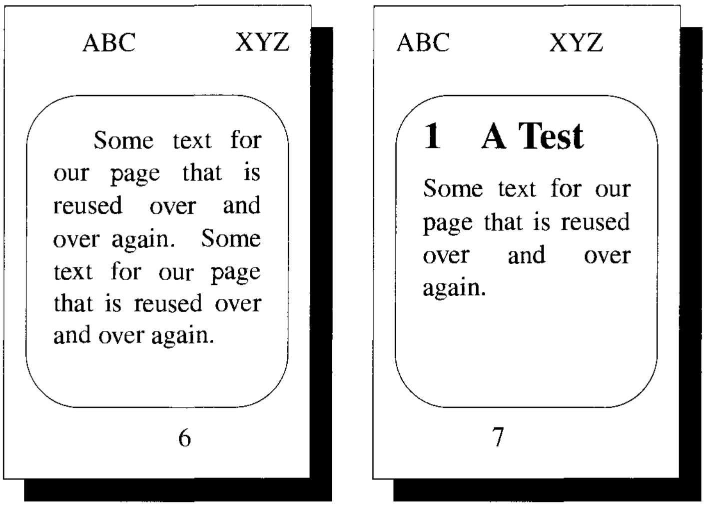



```
\setlength\textwidth{180pt}
\setlength\textheight{7\baselineskip}
\pagestyle{headings}

\usepackage{fancybox}

\newcommand\sample{ Some text for our
 page that is reused over and over again.}
\fancypage
  {\setlength\fboxsep{10pt}\ovalbox}
  {\setlength{\fboxsep}{8pt}%
   \setlength{\shadowsize}{8pt}%
   \shadowbox}
\sample \section{A Test}
\sample\sample
```

10-1-8

Incorrect running headers or footers

Notice that the position of the running header was automatically corrected to fit the extended text width covering the frame. However, this correction works only for standard page styles. If, for example, fancyhdr is used, then the resulting headers and footers will be too small, as this package uses its own method of producing these objects.




```
\usepackage{fancyhdr}
\pagestyle{fancy}
\cfoot{\thepage}
\lhead{ABC} \rhead{XYZ}
% Uncomment next line for
% proper header alignment:
%  \fancyhfoffset[R]{20.8pt}
\usepackage{fancybox}
% \sample as before
\fancypage
 {\setlength\fboxsep{10pt}%
  \ovalbox}
 {\setlength{\fboxsep}{8pt}%
  \setlength{\shadowsize}{8pt}%
  \shadowbox}
\sample\sample
\section{A Test} \sample
```

10-1-9

In the case of fancyhdr, the problem can be corrected by adding an extra offset with `\fancyhfoffset`. The value of `20.8pt` was manually calculated as twice the separation between text and frame (`10pt`) and the width of the frame line (`0.4pt`).

Caveats

The \fancypage declaration is applied to all pages starting with the current one until another \fancypage declaration appears within the document. If you want to add frames only to the current page, use \thisfancypage instead. "Current" in this context means the page under construction when the declaration is first seen by LaTeX, even if that point in the document later ends up on a different page. Thus, it behaves like \pagestyle in this respect. If problems arise, you either have to move the declaration to some earlier or later point in the document or stop LaTeX from looking too far ahead by adding a \pagebreak command somewhere before the declaration.

The other potential problem with the commands \thisfancypage and \fancypage is that they change LaTeX's output routine and, therefore, may not work with other packages that do the same (fancyhdr is an example, though, with some care, both packages can coexist). Also, bad arguments can cause serious errors, which generate uninformative error messages.

\fancyput*(*x*,*y*){*horizontal-material*}

A somewhat more powerful way to add material to every page in fixed locations is provided by the \fancyput declaration. It has a syntax similar to LaTeX's \put command, but requires the specification of dimensions for the x and y coordinates. The origin (0pt,0pt) is one inch from the top and left of the paper. Thus, to put something two inches from the left and three inches from the top, you would specify (1in,-2in).

Some text for our page that is reused over and over again. Some text for our page that is reused over and over again.

DRAFT

1 A Test

Some text for our page that is reused over and over again. Some text for our page that is

10-1-10

```
\usepackage{color,fancybox}
\fancyput(2in,-1.2in)
  {\Huge\bfseries
   \textcolor{blue}{DRAFT}}
% \sample as before
\sample\sample
\section{A Test} \sample \sample
```

The variant form \thisfancyput affects only the current page, analogous to \thisfancypage. If the starred form is used (for either command), then, instead of replacing it, the new material is added to existing material previously inserted with \fancyput or \thisfancyput.

Boxed display environments

The package also predefines boxed versions of the standard LaTeX display environments. The size of the resulting box is determined by the longest line. All environments support an optional argument for positioning the box in relation to the objects on the line; it can be t for top alignment or b for bottom alignment, but the default is to center the box.

The environments Bcenter, Bflushleft, and Bflushright generate a box with the contents centered, flushleft, and flushright, respectively. The example shows all of them in action. Note the use of `\vspace` to ensure that the outer `Bflushleft` box is bottom aligned. Compare this to the examples discussed in Section A.2.2 on page 862.

```
A A A _     B
A A A   B B B
A A     B B B   C C C
_         B B _   C   _
                C C
```

```
\usepackage{fancybox}
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR\begin{Bflushleft}[b]
 \begin{Bflushleft}[t] A A A\\ A A A\\ A A
 \end{Bflushleft}   \HR
 \begin{Bflushright}[t] B\\ B B B\\ B B B\\ B B
 \end{Bflushright}  \par\vspace{0pt}
\end{Bflushleft}    \HR
\begin{Bcenter} C C C\\ C\\ C C\end{Bcenter} \HR
```

10-1-11

`Bitemize`, `Benumerate`, and `Bdescription` implement boxed versions of the `itemize`, `enumerate`, and `description` environments, respectively. The internal implementation uses LaTeX's `tabular` environment, which means that vertical-mode material such as `\vspace` does not work. Instead, the `\item` command takes an optional argument (using parentheses!) to specify extra white space in front of the item. Its usage is shown in the next example.

For math applications, `Beqnarray` produces a boxed environment similar to that created by `eqnarray`, but the equation number always comes out on the right. `Beqnarray*` is like `eqnarray*`, but the generated box is just large enough to hold all the equations. An optional position argument is not supported.

Test:
- First item
- A second one
  on two lines
- A third with extra space

$$\text{Test: } \begin{array}{rcll} y &=& x^2 & (1) \\ a^2+2ab+b^2 &=& (a+b)^2 & (2) \\ \int_0^\infty e^{-ax}dx &=& \frac{1}{a} & (3) \end{array}$$

```
\usepackage{fancybox}
Test: \fbox{\begin{Bitemize}[t]
        \item First item
        \item A second one\\ on two lines
        \item(2pt) A third with extra space
       \end{Bitemize}}
  \par\bigskip
Test: \fbox{\begin{Beqnarray}
                         y & = & x^2       \\
             a^2 + 2ab + b^2 & = & (a + b)^2 \\
     \int_0^\infty e^{-ax} dx & = & \frac{1}{a}
      \end{Beqnarray}}
```

10-1-12

The package also reimplements several commands to typeset verbatim texts. For such applications, however, the fancyvrb package by the same author provides superior interfaces (see Section 3.4.3).

10.1.4 epic—An enhanced `picture` environment

Standard LaTeX provides a `picture` environment that allows you to generate line-style graphics of arbitrary complexity through basic commands for drawing lines, vectors, quarter-circles, and Bézier curves. Thus, creating complex graphics, although possible, requires a lot of manual effort. Most of these picture-drawing commands require explicit specification of coordinates for every *object*. Using higher-level commands can reduce the number of coordinates that need to be manually calculated. Basically, two approaches can be taken to the design of such commands:

- A set of objects can be selected so that the entire set can be plotted by specifying one or two coordinate pairs—the `\shortstack` command falls under this approach.
- Commands are provided that will do most of the computations internally and require only simple coordinate pairs to be specified—the `\multiput` command is an example of this approach.

The obvious advantage of using commands that implement these approaches is not only that they are easier to specify initially, but any subsequent modification to the layout requires minimal recalculations.

The frequently used primitive command `\line` has severe limitations and drawbacks. Its arguments are very nonintuitive and require extensive calculations. Often the thought process in writing a `\line` command involves several steps:

1. Calculating the coordinates of the two end points
2. Calculating the horizontal and vertical distance
3. Translating these distances into an (x, y) pair for specifying a slope and a horizontal distance for specifying the length of the line
4. Determining whether the desired slope is available and, if not, repeating steps 1 through 3 until a satisfactory slope is achieved

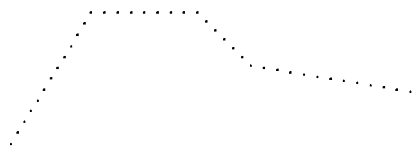
This mechanism is very cumbersome. Moreover, the length of the shortest available line at different slopes is not the same due to the way that the `\line` command is implemented. To overcome these difficulties, the epic package (by Sunil Podar) provides a powerful high-level user interface to the `picture` environment [139]. Its main aim is to reduce the amount of manual calculations required to specify the layout of *objects*. In this way, the epic package makes it possible to produce sophisticated pictures with less effort than before.

High-level line commands

The package introduces a number of powerful line-drawing commands, while at the same time providing a simpler syntax. In particular, these commands take only the coordinates of the end points, thus eliminating the other steps involved in specifying a line.

\dottedline[*dotchar*]{*dotgap*}(x_1, y_1)(x_2, y_2)...(x_n, y_n)

The \dottedline command connects the specified points by drawing a dotted line between each pair of coordinates. At least two points must be defined. The dotted line is drawn with an inter-dot gap as specified in the mandatory argument *dotgap* (in \unitlength). Because the number of dots to be plotted must be an integer, the inter-dot gap may not come out exactly as specified.

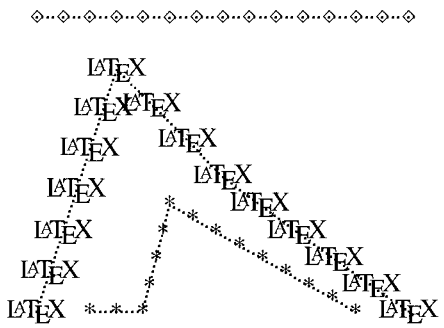


```
\usepackage{epic}\setlength{\unitlength}{1pt}
\begin{picture}(150,80)(0,0)
 \dottedline{2}(0,00)(50,20)(100,80)(150,0)
 \thicklines
 \dottedline{5}(0,0)(30,50)(70,50)(90,30)(150,20)
\end{picture}
```

10-1-13

By default (i.e., if no optional *dotchar* argument is used), \dottedline plots tiny squares, produced internally by the \picsquare command. The size of the squares depends on the current setting of the \thinlines, \thicklines, or \linethickness command. In fact, most of the epic commands internally use \picsquare for plotting lines.

By using the optional *dotchar* argument, you can plot any object along the line specified by the coordinates. Note that some characters like "*" in the Roman font do not come out centered, although most other characters and objects do.

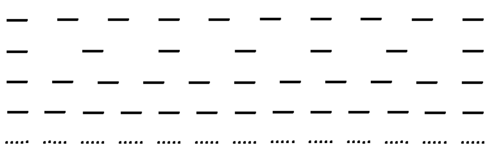


```
\usepackage{epic}
\setlength{\unitlength}{1pt}   \thicklines
\begin{picture}(140,110)(0,0)
  \dottedline             {2}(0,110)(140,110)
  \dottedline[$\diamond$]{10}(0,110)(140,110)
  \dottedline    {2}(20,0)(40,0)(50,40)(120,0)
  \dottedline[*]{10}(20,0)(40,0)(50,40)(120,0)
  \dottedline          {2}(0,0)(30,90)(70,50)(140,0)
  \dottedline[\LaTeX]{20}(0,0)(30,90)(70,50)(140,0)
\end{picture}
```

10-1-14

\dashline[*stretch*]{*dashlength*}[*dashdotgap*](x_1, y_1)(x_2, y_2)...(x_n, y_n)

The \dashline command connects the specified points by drawing a dashed line between each pair of coordinates. At least two points must be specified. Internally, each dash is constructed using the \dottedline command. The mandatory parameter *dashlength* determines the length of each dash, and the optional argument *dashdotgap* gives the gap between the dots that are used to construct the dash, both in `\unitlength` terms. By default, a solid-looking dash is constructed.



10-1-15

```
\usepackage{epic}
\setlength{\unitlength}{1mm}
\begin{picture}(70,22)(0,-2)
\dashline{3}[0.7](0,20)(63,20)
\thicklines
\dashline{3}(0,16)(63,16)
\dashline[-30]{3}(0,12)(63,12)
\dashline[+15]{3}(0,8)(63,8)
\dashline[+30]{3}(0,4)(63,4)
\dashline[+30]{3}[0.7](0,0)(63,0)
\end{picture}
```

In the definition of the `\dashline` command, the optional *stretch* parameter must be an integer between -100 and ∞. It indicates the percentage by which the number of dashes is "stretched" or increased (*stretch* > 0) or is "shrunk" or reduced (*stretch* < 0). If *stretch* is zero, the minimum number of dashes compatible with an approximately equal spacing relative to the empty space between the dashes is used. The idea behind the *stretch* percentage parameter is that if several dashed lines of different lengths are being drawn, then all dashed lines with identical *stretch* values will have a similar visual appearance. The default settings for the *stretch* percentage can be changed by redefining the command `\dashlinestretch`:

```
\renewcommand\dashlinestretch{-50} % Only integers permitted
```

Its value defines the increase or reduction that will be applied to all subsequent `\dashline` commands except for those where the *stretch* parameter is explicitly specified as the first optional argument.

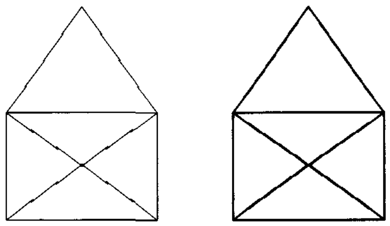
`\drawline[`*stretch*`]`$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

The `\drawline` command connects the given points by drawing a line between each pair of coordinates using line segments of the closest slope available in the line fonts of LaTeX. A minimum of two points must be specified. Only a finite number of slopes are available in the line segment fonts, so unavailable slopes are produced by repeatedly using very short line segments of a nearby slope. As a consequence, some lines may appear jagged (in the next example all sloped lines show this effect). This is the price you must pay for being allowed to implicitly specify lines of any slope. However, the problem vanishes if the eepic package is used in addition to epic.

Unwanted jagged lines

A `\drawline` command can generate thick or thin lines depending on the setting of the `\thinlines` or `\thicklines` parameters in effect. These are the only two thicknesses available for such lines.

The optional *stretch* parameter is similar to the one described for the `\dashline` command. If *stretch* is zero, the result is the minimum number of dashes required to make the line appear solid, with each dash being "connected" at the ends. If *stretch* is greater than zero, more dashes are used in constructing the line, giving a less jagged appearance (compare the two houses in the example).



```
\usepackage{epic} \setlength{\unitlength}{2mm}
\begin{picture}(25,14)
  \drawline(0,0)(0,7)(5,14)(10,7)
           (0,7)(10,0)(0,0)(10,7)(10,0)

  \thicklines
  \drawline[70](15,0)(15,7)(20,14)(25,7)
              (15,7)(25,0)(15,0)(25,7)(25,0)
\end{picture}
```

10-1-16

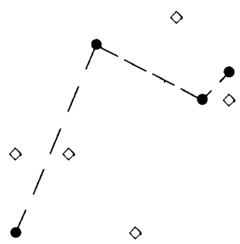
As with the `\dashlinestretch` parameter and the `\dashline` command, the parameter `\drawlinestretch` allows you to set the default value for the *stretch* percentage parameter of the `\drawline` command.

Plotting scientific data

When presenting scientific data, it is often desirable to produce graphs that show obtained (two-dimensional) data sets in relation to each other. One representation strategy is to plot one set of experimentally obtained data points using a certain type of graphical representation (e.g., filled circles) and another using some different symbol (e.g., diamonds). For further clarification you might want to join the individual data points with some kind of line, perhaps using different types of "lines" to help the reader distinguish between the resulting curves.

One way to achieve this result is to plot the experimental results using a sequence of basic `\put` statements, followed by a `\dottedline`, `\dashline`, or `\drawline` command, that connects the data points. In other words, you specify the coordinates twice. To facilitate this process, epic offers the three environments `dottedjoin`, `dashjoin`, and `drawjoin` corresponding to the above commands and accepting the same optional and mandatory arguments. These environments use the new command `\jput` (join and put), which is identical to the regular `\put` command of LaTeX except that it can be used inside these three environments only. All objects put within the scope of any of the three environments via a `\jput` command are, in addition to being plotted, joined by lines of their respective type. It is up to the user to center the objects at the plotted points.

An instance of any of the three `..join` environments defines a separate "curve"; hence, every set of points belonging to a different "curve" should be enclosed in a separate ..join environment. The prime motivation for designing the ..join environments was to allow for plotting graphs that use different types of curves and dissimilar lines.



10-1-17

```
\usepackage{epic} \setlength{\unitlength}{1pt}
\newcommand\cb{\makebox(0,0){$\bullet$}}
\newcommand\cd{\makebox(0,0){$\diamond$}}
\begin{picture}(80,80)
  \begin{dashjoin}[30]{10}
    \jput(0,0){\cb}\jput(30,70){\cb}\jput(70,50){\cb}\jput(80,60){\cb}
  \end{dashjoin}
  \begin{dottedjoin}{5}
    \jput(0,30){\cd}\jput(20,30){\cd}\jput(45,0){\cd}\jput(60,80){\cd}
    \jput(80,50){\cd}
  \end{dottedjoin}
\end{picture}
```

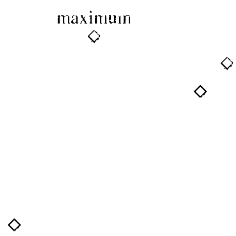
Another way to produce graphs that is offered by the epic package is through the \putfile{*file*}{*object*} command. It is similar to LaTeX's \put command, except that the x and y coordinates required by the \put command are read from an external file and the same *object* is plotted at each of those coordinates. This command is provided because TeX lacks the capability of doing floating-point arithmetic, which is required if you wish to plot a parametric curve different from a straight line. The coordinates of points on such curves can easily be generated by a program in some computer language and subsequently read in by TeX. The external file must contain the (x, y) coordinate pairs, one pair per line, with a space between the two coordinates. The % is available as a comment character, but you should leave at least one space following the y entry if a comment appears on the same line as data because a % masks the newline character.

Loading externally generated graphic data

For example, to plot a smooth curve along a set of coordinates, you can use the following procedure:

1. Create a file with the x, y coordinates of the data points, which you might call plot.data, for example.
2. If you wish, smooth the data.
3. Place the following code inside a picture environment in your LaTeX file: \putfile{*plot.data*}{\picsquare}

As the command name indicates, \putfile uses \put and not \jput. This choice is unfortunate, as it means that using \putfile inside one of the ..join environments will plot objects at the coordinates but not connect them, even though there is technically nothing to prevent this connection. There is, however, a small trick you can use if you are interested in creating such linkage: ensure that \put always executes \jput inside your pictures. Because \jput behaves exactly like LaTeX's \put command if used outside the ..join environments, there is no harm in making this a global substitution. This approach is used in the next example.




```
\usepackage{epic}    \renewcommand\put{\jput} % <- always use \jput
\begin{filecontents}{test.put}
0 0                                     % sample data in external file
30 70                                   % note that coordinates are
70 50                                   % separated by a space
80 60
\end{filecontents}
\newcommand\cd{\makebox(0,0){$\diamond$}}
\begin{picture}(80,80)
 \begin{dashjoin}{6}[2] \putfile{test.put}{\cd} \end{dashjoin}
 \put(30,75){\makebox(0,0)[b]{\scriptsize maximum}}
\end{picture}
```

10-1-18

Placing objects at regular intervals

What is missing in the example graphs so far are labeled axes. The epic doesn't offer off-the-shelf commands to do the full job, but with `\multiputlist` and `\grid` it offers tools that can help you with the more tedious tasks.

`\multiputlist(`x`,`y`)(`Δx`,`Δy`)[`*pos*`]{`*item*$_1$`,`*item*$_2$`,`*item*$_3$`,...,`*item*$_n$`}`

This command is a variant of LaTeX's `\multiput` command, which allows the *same* object to be placed at regularly spaced coordinates. The `\multiputlist` command is similar, but permits the objects to be *different*. When the `\multiputlist` command is executed, the objects to be "put" are picked up from the *list of items*, as the coordinates are incremented. (The first item goes in position 1, the second item in position 2, and so on.) For example, you can plot numbers along the x-axis in a graph by specifying

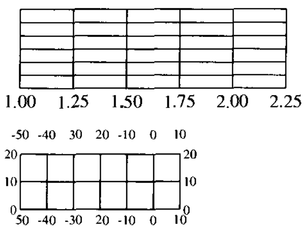
```
\multiputlist(0,0)(10,0){1.00,1.25,1.50,1.75,2.00}
```

The objects in the list can be virtually anything, including `\makebox`, `\framebox`, or math characters. This command enforces a certain regularity and symmetry on the layout of the various objects in a picture.

`\grid(`*width*`,`*height*`)(`Δ*width*`,`Δ*height*`)[`*initial-X-int*`,`*initial-Y-int*`]`

The `\grid` command makes a grid of dimensions *width* units by *height* units. Vertical lines are drawn at intervals of $\Delta width$ and horizontal lines at intervals of $\Delta height$. When the third (optional) argument is specified, the borders of the grid will be labeled with numbers whose starting values are the integer numbers *initial-X-int* and *initial-Y-int*, respectively. They will be incremented by $\Delta width$ and $\Delta height$ along the axes.

The \grid command produces a box. Therefore, it must be \put at the required coordinates. For example:



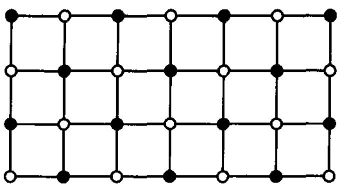

```
\usepackage{epic}
\begin{picture}(100,60)
  \put(0,45){\grid(100,30)(20,5)}
  \scriptsize % used to influence the size of the numbers
  \multiputlist(0,40)(20,0){1.00,1.25,1.50,1.75,2.00,2.25}

  \put(0,0){\tiny\grid(60,20)(10,10)[-50,0]}
\end{picture}
```

10-1-19

If you need more flexibility than that offered by \grid for producing a regular two-dimensional structure, then \matrixput might offer the answer.

\matrixput$(x,y)(\Delta x_1,\Delta y_1)$\{$n_1$\}$(\Delta x_2,\Delta y_2)$\{$n_2$\}\{*object*\}

This command is the two-dimensional equivalent of the primitive LaTeX command \multiput. It is more efficient, however, to use \matrixput than multiple \multiput statements. This command is especially useful for drawing pictures where a pattern is repeated at regular intervals in two dimensions.



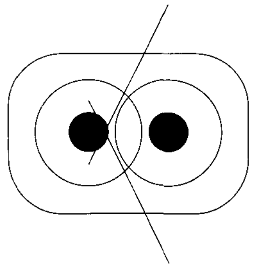
```
\usepackage{epic} \setlength{\unitlength}{2pt}
\begin{picture}(62,32)    \thicklines
  \matrixput(0,0)(10,0){7}(0,10){4}{\circle{2}}
  \matrixput(10,0)(20,0){3}(0,20){2}{\circle*{2}}
  \matrixput(0,10)(20,0){4}(0,20){2}{\circle*{2}}
  \matrixput(1,0)(10,0){6}(0,10){4}{\line(1,0){8}}
  \matrixput(0,1)(10,0){7}(0,10){3}{\line(0,1){8}}
\end{picture}
```

10-1-20

10.1.5 eepic—Extending the epic package

LaTeX provides a basic but limited picture-drawing capability, which is extended by commands for drawing solid lines, dotted lines, dashed lines, and new environments suitable for plotting graphs of the epic package (described in the previous section). However, epic inherits many of LaTeX's limitations in picture drawing. As a result, some of the functions take a long time to accomplish or the output is not of very high quality. In LaTeX, special fonts are used to draw lines and circles. For this reason only lines with certain slopes are supported and only a limited set of diameters is available when drawing circles, ovals, or disks.

The following example shows some of these limitations. Here, the circle and disk on the left are too small (without producing any warning) and the \line commands produce errors because the required slope is not available. Loading epic does not help in this case.



```
\usepackage{epic}
\begin{picture}(0,0)
  \put(0,0) {\circle{80}} \put(0,0) {\circle*{24}}
  \put(30,0){\circle{40}} \put(30,0){\circle*{16}}
  \put(15,0){\oval(90,60)}
  \put(0,12){\line(15,-2){30}}\put(0,-12){\line(15,2){30}}
\end{picture}
```

10-1-21

Compare this result to Example 10-1-22 on the next page, which shows the correct output—it is strikingly different.

At the end of the 1980s, the pic programming language was developed to provide a "natural language" method of describing simple pictures and graphs (see [77]). A preprocessor, like GNU's gpic, can translate these graphics commands into output that the UN*X formatter, troff, understands. More interestingly for us, it can also generate TeX `\special` commands, which many `dvi` driver programs support. For instance, the dvips `dvi`-to-PostScript translator, described in Section 10.4.2, can interpret these commands.

The eepic package, written by Conrad Kwok, is an extension of both LaTeX and epic that overcomes some of the limitations in LaTeX, epic, and gpic by generating gpic `\specials` using TeX commands. Because eepic is a superset of epic, you can use it to process any picture that relies on epic commands and get better-looking output.

eepic's reimplementation of LaTeX commands

The extensions in eepic allow users to draw lines having any slope and to draw circles of any size. However, the limitation of slopes for vectors remains the same. Thus, the only slopes that can be handled are of the form x/y, where x and y are integers in the range $[-4, 4]$.

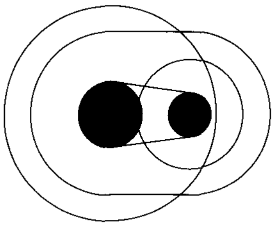
`\line(`*x*`,`*y*`){`*length*`}`

The syntax of the `\line` command is the same in eepic as in LaTeX. Now, however, x and y can be any integers acceptable to TeX. Furthermore, there is no longer a lower limit for the *length* parameter (about 3.5 mm in standard LaTeX).

`\circle{`*diameter*`}` `\circle*{`*diameter*`}` `\oval(`*x*`,`*y*`)[`*part*`]`

The syntax for drawing hollow and filled circles, `\circle` and `\circle*`, is the same as that in LaTeX. Now, however, the *diameter* parameter can be any number acceptable to TeX, and a circle with a diameter of (exactly) the specified value will be drawn. The \oval command has been modified so that the maximum diameter of the quarter-circles at the corners can be set to any value by setting the variable \maxovaldiam to the desired TEX dimension (default 40pt).

The following example repeats Example 10-1-21 on the facing page, except that now eepic has been loaded and \maxovaldiam has been used. All elements appear as specified in the revised example.



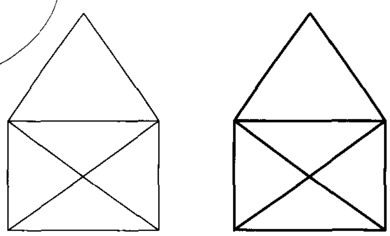
```
\usepackage{eepic} \setlength\maxovaldiam{60pt}
\begin{picture}(0,0)
  \put(0,0) {\circle{80}} \put(0,0) {\circle*{24}}
  \put(30,0){\circle{40}} \put(30,0){\circle*{16}}
  \put(15,0){\oval(90,60)}
  \put(0,12){\line(15,-2){30}}\put(0,-12){\line(15,2){30}}
\end{picture}
```

10-1-22

eepic's reimplementation of epic commands

The epic package generates standard dvi files and requires the presence of only the standard LaTeX fonts. The eepic package, as an extension to epic, offers better line-drawing output, provides faster operation, and requires less memory. It reimplements the \drawline, \dashline, and \dottedline commands (see page 601) and the corresponding ..join environments, dashjoin, dottedjoin, and drawjoin (see page 604).

Compare the diagonal lines in the following example with those in Example 10-1-16 on page 604. Note that when eepic is loaded in conjunction with epic it smoothes the result of any line-drawing command. Both packages must be loaded in the right order.



```
\usepackage{epic,eepic} \setlength{\unitlength}{2mm}
\begin{picture}(25,14)
  \drawline(0,0)(0,7)(5,14)(10,7)
           (0,7)(10,0)(0,0)(10,7)(10,0)
  \thicklines
  \drawline[70](15,0)(15,7)(20,14)(25,7)
               (15,7)(25,0)(15,0)(25,7)(25,0)
\end{picture}
```

10-1-23

The eepic package also introduces a number of new commands. Apart from the \path command, these commands do not have equivalents in LaTeX and epic. The end of this section discusses portability issues as they relate to these packages.

\allinethickness{*dimension*}   \Thicklines

The \allinethickness command sets the line thickness of all line-drawing commands, including lines in slopes, circles, ellipses, arcs, ovals, and splines.

After issuing `\Thicklines`, the thickness of all subsequently drawn lines will be about 1.5 times greater than that with `\thicklines`.

`\path`$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

The `\path` command is a fast version of the `\drawline` command. The optional *stretch* argument of the latter command is not allowed, so `\path` draws only solid lines. This command is mainly used for drawing complex paths.

`\spline`$(x_1,y_1)(x_2,y_2)\ldots(x_n,y_n)$

The `\spline` command draws a Chaikin's curve that passes through only the first and last points. All other points act as control points only.

`\ellipse`{*x-diameter*}{*y-diameter*}   `\ellipse*`{*x-diameter*}{*y-diameter*}

In analogy to the `\circle` and `\circle*` commands, the `\ellipse` and `\ellipse*` commands draw a hollow or filled ellipse using the specified *x-diameter* and *y-diameter* parameters.

`\arc`{*diameter*}{*start-angle*}{*end-angle*}

The `\arc` command draws a circular arc. The first parameter, *diameter*, is given in `\unitlength` terms. Both *start-angle* and *end-angle* are in radians; *start-angle* must lie within the interval $[0, \frac{\pi}{2}]$, and *end-angle* can be any value between *start-angle* and *start-angle* + 2π. Arcs are drawn clockwise, with the angle 0 pointing to the right on the paper.

`\filltype`{*area-fill-type*}

The `\filltype` command specifies the type of area fill for the `\circle*` and `\ellipse*` commands. The instruction itself does not draw anything, but merely changes the interpretation of `*` in the two commands specified above. Possible values for *area-fill-type* are `black` (default), `white`, and `shade`. For example, you can change the area fill type to white with `\filltype{white}`.

The eepic package is not necessarily available at all LaTeX sites or, even if it is available, it may not be supported by the chosen output device. To avoid the portability problems that can arise from its use, and at the same time take advantage of eepic's more precise printout, take the following precautions:

Portability issues

- Do not use `\line` commands, but use `\drawline` instead. The `\line` command in LaTeX supports only a limited set of slopes.
- Do not use the `\arc` command. Use the command `\spline` if a complex curve is really necessary.

- Avoid using solid or small inter-dot gaps in drawing long dashed lines, as these need a lot of TeX memory in the original epic implementation. Use the `\drawline` command with negative stretch to draw dashed lines.

Emulating the eepic *commands*

If your installation does not support eepic but you have to print your document, then you should use the eepic emulation macros defined with the eepicemu package. The extended commands are emulated in the following ways:

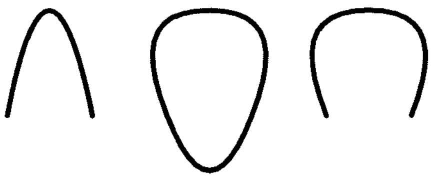
- Circles larger than 40pt are drawn using `\oval`.
- Ellipses are drawn using `\oval`.
- Arcs generate a warning but are ignored otherwise.
- Splines are approximated with `\drawline`.
- `\path` is substituted by `\drawline`.
- `\Thicklines` is substituted by `\thicklines`.
- `\allinethickness` is substituted by `\thicklines` and `\linethickness`.

Because the eepic package redefines several commands of the epic package, the eepic package declaration must follow the epic package declaration. Although not strictly necessary, it is good practice to always include epic when using eepic commands. In any case, the eepic emulation package eepicemu will work only when both are specified.

10.1.6 Special-purpose languages

Building on LaTeX's `picture` environment, possibly extended with the epic and eepic packages, several package authors have implemented high-level user interfaces intended ~~to make~~ entering graphical information more straightforward and less error prone by adopting a syntax that is more familiar to the end user in a particular application domain. Some of the systems are quite complex (*The Graphics Companion* [57] describes several of them in detail). In this section we merely give a flavor of what is possible in this area by showing a few short examples.

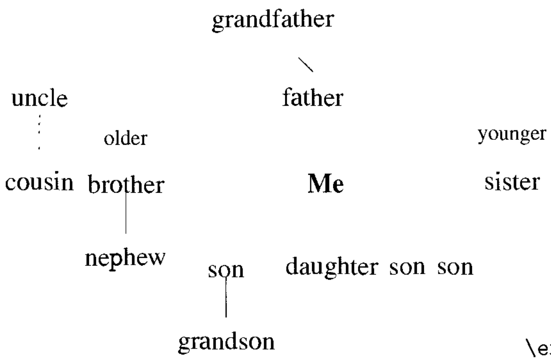
If you do not have access to a drawing package but need to include a few continuously sloping curves, the curves package written by I. L. Maclaine-cross offers some intriguing features. It allows you to vary curve thickness over a large range, to control end slopes, and to specify closed curves with continuous slopes. It can also build large circles and circular arcs with `\arc`, providing independent scaling of curve abscissa and ordinates to fit graphs. Furthermore, it offers affine scaling for making arcs or circles become elliptical and it supports symbols and dash patterns. In the simple example that follows, `\curve` draws a curve through the specified coordinate pairs, `\closecurve` draws a closed curve with continuous tangents at all points, and `\tagcurve` generally acts like `\curve` except that the first and last segments are not drawn.



```
\usepackage{curves}
\setlength{\unitlength}{0.4pt}
\linethickness{0.7mm}
\begin{picture}(400,110)(-10,0)
   \curve(0,0, 40,100, 80,0)
   \closecurve(150,0, 190,100, 230,0)
   \tagcurve(380,0, 300,0, 340,100, 380,0, 300,0)
\end{picture}
```

10-1-24

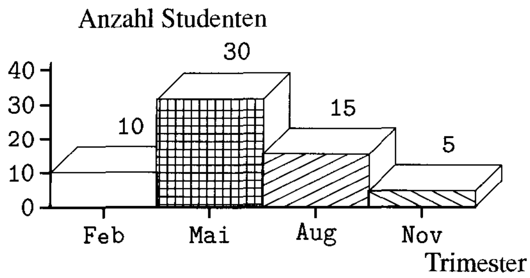
Hideki Isozaki's ecltree package allows you to draw simple tree structures. It offers a `bundle` environment for labeling a top node, which can contain one or more down nodes defined by `\chunk` commands, whose optional argument can be used to add comments on a line. The `\drawwith` command allows you to control the line style by specifying as an argument one of epic's line-drawing commands (described in Section 10.1.4). The `bundle` environment and `\chunk` commands can be nested, as shown in the following LaTeX code.




```
\usepackage{epic,eepic,ecltree}
\begin{bundle}{grandfather}
 \chunk{\begin{bundle}{uncle\strut}
        \chunk{cousin}\drawwith{\dottedline{3}}
       \end{bundle}}
 \chunk{\begin{bundle}{father\strut}
        \chunk[\footnotesize older]{%
          \begin{bundle}{brother}
            \chunk{nephew}
          \end{bundle}}
        \chunk{\begin{bundle}{\textbf{Me}\strut}
                \chunk{\begin{bundle}{son}
                        \chunk{grandson}
                      \end{bundle}}
                \chunk{daughter}\chunk{son}\chunk{son}
              \end{bundle}}
        \chunk[\footnotesize younger]{sister}
       \end{bundle}}
\end{bundle}
```

10-1-25

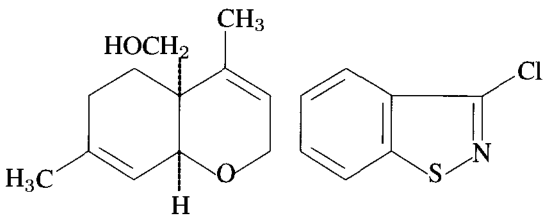
The bar package was written by Joachim Bleser and Edmund Lang to produce bar charts. A `barenv` environment encloses the data defining a bar chart. Each data point is specified using a `\bar` command, whose two mandatory arguments give the ordinate of the entry and the hatching type. The package also offers quite a few `\set...` commands to fine-tune the presentation of the information, as shown in the example that follows.




```
\usepackage{epic,eepic,bar}
\begin{barenv}
\setdepth{10}% 3-D effect
\setstretch{1.4}% stretch y-dimension
\setnumberpos{up}% numbers above bars
\setxvaluetyp{month}% (German) months on x-axis
\setxaxis{2}{12}{3}\setxname{Trimester}
\setyaxis{0}{40}{10}\setyname{Anzahl Studenten}
\bar{10}{1} \bar{30}{4}
\bar{15}{6} \bar{5}{7}
\end{barenv}
```

10-1-26

As already stated, much more complex structural data can be entered in a convenient way by using a dedicated package. One example is Shinsaku Fujita's XΥMTeX bundle for drawing chemical diagrams (see [48, 49] or [57, Chapter 6]). By using command names inspired by standard nomenclature known to practitioners in the field, complex formulas can be entered simply. In the following example, we use the hetarom subpackage, designed for specifying the structure of vertical heterocyclic compounds.




```
\usepackage{eepic,hetarom}
\decaheterov[af]{4==O}
  {1==CH$_3$;6==H$_3$C;9A==H;%
   {{10}A}==\lmoiety{HOCH$_2$}}
\hspace*{-15mm}
\nonaheterov[bjge]{1==S;2==N}{3==Cl}
```

10-1-27

10.2 LaTeX's device-dependent graphics support

Since the introduction of LaTeX 2_ε in 1994, LaTeX has offered a uniform syntax for including every kind of graphics file that can be handled by the different drivers. In addition, all kinds of graphic operations (such as resizing and rotating) as well as color support are available.

These features are not part of the LaTeX 2_ε kernel, but rather are loaded by the standard, fully supported color, graphics, and graphicx extension packages. As the TeX program does not have any direct methods for graphic manipulation, the packages have to rely on features supplied by the "driver" used to print the dvi file. Unfortunately, not all drivers support the same features, and even the internal method of accessing these extensions varies among drivers. Consequently, all of these packages take options such as dvips to specify which external driver is being used. Through this method, unavoidable device-dependent information is localized in a single place, the preamble of the document.

The packages graphics and graphicx can both be used to scale, rotate, and reflect LaTeX material, or to include graphics files prepared with other programs. The difference between the two is that graphics uses a combination of macros with a "standard" or TeX-like syntax, while the "extended" or "enhanced" graphicx package presents a key/value type of interface for specifying optional parameters to the `\includegraphics` and `\rotatebox` commands.

10.2.1 Options for graphics and graphicx

When using LaTeX's graphics packages, the necessary space for the typeset material after performing a file inclusion or applying some geometric transformation is reserved on the output page. It is, however, the task of the *device driver* (e.g., dvips, xdvi, dvipsone) to perform the actual inclusion or transformation in question and to show the correct result. As different drivers require different code to carry out an action like rotation, one has to specify the target driver as an option to the graphics packages—for example, option `dvips` if you use one of the graphics packages with Tom Rokicki's dvips program, or option `textures` if you use one of the graphics packages and work on a Macintosh using Blue Sky's Textures program.

Some drivers, such as previewers, are incapable of performing certain of the desired functions. Hence, they may display the typeset material so that it overlaps with the surrounding text. Table 10.1 on the facing page shows the drivers currently supported and their possible limitations. Support for other drivers is added occasionally, so it is worth checking the online documentation of the package for a driver not listed in this table.

The driver-specific code is stored in files with the extension `.def`—for example, `dvips.def` for the PostScript driver dvips. As most of these files are maintained by third parties, the standard LaTeX distribution contains only a subset of the available files and not necessarily the latest versions. While there is usually no problem if LaTeX is installed as part of a full TeX installation, you should watch out for incompatibilities if you update the LaTeX graphics packages manually.

Setting a default driver

It is also possible to specify a default driver using the `\ExecuteOptions` declaration in the *configuration* file `graphics.cfg`. For example, the declaration `\ExecuteOptions{emtex}` makes the emTeX drivers become the default. In this case the graphics packages pick up the driver code for the emTeX TeX system on a PC if the package is called without a driver option. These days most TeX installations are distributed with a ready-to-use `graphics.cfg` file.

In addition to the driver options, the packages support some options controlling which features are enabled (or disabled):

`draft` Suppress all "special" features, such as including external graphics files in the final output. The layout of the page will not be affected, because LaTeX still reads the size information concerning the bounding box of the external material. This option is of particular interest when a document is under development and you do not want to download the (often huge)

Option	*Author of Driver*	*Features*
dvips	T. Rokicki	All functions
dvialw	N. Beebe	File inclusion with scaling only
dvipdf	S. Lesenko	All functions
dvilaser	Arbortext	File inclusion with scaling only
dvipsone	Y&Y	All functions
dvitops	J. Clark	All functions, but no nested rotations
dviwin	H. Sendoukas	File inclusion
dviwindo	Y&Y	All functions
dvi2ps	original	File inclusion with scaling only
emtex	E. Mattes	File inclusion only, but no scaling
ln	B. H Kelly	File inclusion for DEC's LN03 printer
oztex	A. Trevorrow	File inclusion, color, rotation
pdftex	Hán Thế Thánh	All functions
pctexps	PCTeX	File inclusion, color, rotation
pctexwin	PCTeX	File inclusion, color, rotation
pctex32	PCTeX	All functions
pctexhp	PCTeX	File inclusion only
psprint	A. Trevorrow	File inclusion only
pubps	Arbortext	Rotation, file inclusion
truetex	Kinch	Graphics inclusion and some color
tcidvi	Kinch	TrueTeX with extra support for Scientific Word
textures	Blue Sky	All functions for Textures

Table 10.1: Overview of color and graphics capabilities of device drivers

graphics files each time you work on it. When `draft` mode is activated, the picture is replaced by a box of the correct size containing the name of the external file.

`final` The opposite of `draft`. This option can be useful when, for instance, "draft" mode was specified as a global option with the `\documentclass` command (e.g., for showing overfull boxes), but you do not want to suppress the graphics as well.

`hiresbb` In PostScript files look for bounding box comments that are of the form `%%HiResBoundingBox` (which typically have real values) instead of the standard `%%BoundingBox` (which should have integer values). With the graphicx package, this and the previous options are also available locally for individual `\includegraphics` commands.

`hiderotate` Do not show the rotated material (for instance, when the previewer cannot rotate material and produces error messages).

`hidescale` Do not show the scaled material (for instance, when the previewer does not support scaling).

```
%!PS-Adobe-2.0
%%BoundingBox:100 100 150 150
100 100     translate % put origin at 100 100
  0   0     moveto    % define current point
 50  50     rlineto   % trace diagonal line
 50 neg 0   rlineto   % trace horizontal line
 50  50 neg rlineto   % trace other diagonal line
stroke                % draw (stroke) the lines
  0   0     moveto    % redefine current point
/Times-Roman findfont % get Times-Roman font
 50         scalefont % scale it to 50 big points
            setfont   % make it the current font
(W) show              % draw an uppercase W
```

Figure 10.1: The contents of the file `w.eps`

10.2.2 The `\includegraphics` syntax in the graphics package

With the graphics package, an image file can be included by using the following command:

`\includegraphics*`[*llx*, *lly*] [*urx*, *ury*] {*file*}

If the [*urx*, *ury*] argument is present, it specifies the coordinates of upper-right corner of the image as a pair of TEX dimensions. The default units are big (PostScript) points; thus, `[1in,1in]` and `[72,72]` are equivalent. If only one optional argument is given, the lower-left corner of the image is assumed to be located at `[0,0]`. Otherwise, [*llx*, *lly*] specifies the coordinates of that point. Without optional arguments, the size of the graphic is determined by reading the external *file* (containing the graphics itself or a description thereof; see below).

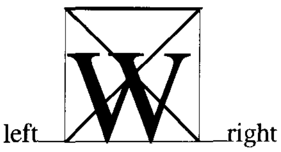
The starred form of the `\includegraphics` command "clips" the graphics image to the size of the specified bounding box. In the normal form (without the *), any part of the graphics image that falls outside the specified bounding box overprints the surrounding text.

The examples in the current and next sections use a small PostScript program (in a file `w.eps`) that paints a large uppercase letter "W", and a few lines. Its source is shown in Figure 10.1. Note the BoundingBox declaration, which stipulates that the image starts at the point `100, 100` (in big points), and goes up to `150, 150`; that is, its natural size is 50 big points by 50 big points.

In the examples we always embed the `\includegraphics` command in an `\fbox` (with a blue frame and zero `\fboxsep`) to show the space that LaTeX reserves for the included image. In addition, the baseline is indicated by the horizontal rules produced by the `\HR` command, defined as an abbreviation for `\rule{1em}{0.4pt}`.

The first example shows the inclusion of the `w.eps` graphic at its natural size. Here the picture and its bounding box coincide nicely.

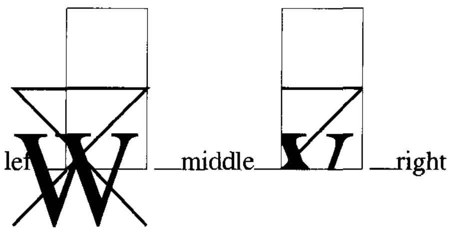
10-2-1




```
\usepackage{graphics,color}
\newcommand\HR{\rule{1em}{0.4pt}}
\newcommand\bluefbox[1]{\textcolor{blue}{%
   \setlength\fboxsep{0pt}\fbox{\textcolor{black}{#1}}}}
left\HR \bluefbox{\includegraphics{w.eps}}\HR right
```

Next, we specify a box that corresponds to a part of the picture (and an area outside it) so that some parts fall outside its boundaries, overlaying the material surrounding the picture. If the starred form of the command is used, then the picture is clipped to the box, as shown on the right.

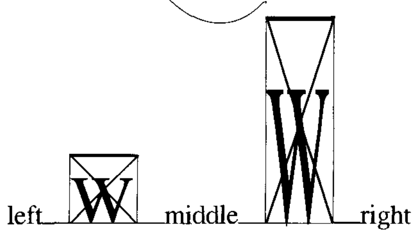
10-2-2



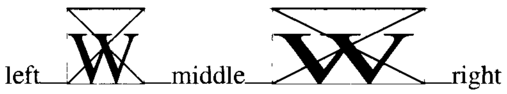

```
\usepackage{graphics,color}
% \bluefbox and \HR as before
left\HR
 \bluefbox{\includegraphics
         [120,120][150,180]{w.eps}}
\HR middle\HR
 \bluefbox{\includegraphics*
         [120,120][150,180]{w.eps}}
\HR right
```

In the remaining examples we combine the `\includegraphics` command with other commands of the graphics package to show various methods of manipulating an included image. (Their exact syntax is discussed in detail in Section 10.3.) We start with the `\scalebox` and `\resizebox` commands. In both cases we can either specify a change in one dimension and have the other scale proportionally, or specify both dimensions to distort the image.

10-2-3



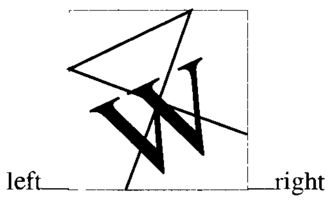

```
\usepackage{graphics,color}
% \bluefbox and \HR as before
left\HR
  \bluefbox{\scalebox{.5}{%
    \includegraphics{w.eps}}}%
\HR middle\HR
  \bluefbox{\scalebox{.5}[1.5]{%
    \includegraphics{w.eps}}}%
\HR right
```




```
\usepackage{graphics,color}
% \bluefbox and \HR as before
left\HR
  \bluefbox{\resizebox{10mm}{!}{%
    \includegraphics{w.eps}}}%
\HR middle\HR
  \bluefbox{\resizebox{20mm}{10mm}{%
    \includegraphics{w.eps}}}%
\HR right
```

10-2-4

Adding rotations makes things even more interesting. Note that in comparison to Example 10-2-1 on the preceding page the space reserved by LaTeX is far bigger. LaTeX "thinks" in rectangular boxes, so it selects the smallest size that can hold the rotated image.




```
\usepackage{graphics,color}
% \bluefbox and \HR as before
left\HR
  \bluefbox{\rotatebox{25}{%
    \includegraphics{w.eps}}}%
\HR right
```

10-2-5

10.2.3 The `\includegraphics` syntax in the `graphicx` package

The extended graphics package graphicx also implements `\includegraphics` but offers a syntax for including external graphics files that is somewhat more transparent and user-friendly. With today's TeX implementations, the resultant processing overhead is negligible, so we suggest using this interface.

`\includegraphics*` [*key/val-list*] {*file*}

The starred form of the command exists only for compatibility with the standard version of `\includegraphics`, as described in Section 10.2.2. It is equivalent to specifying the `clip` key.

The *key/val-list* is a comma-separated list of *key*=*value* pairs for keys that take a value. For Boolean keys, specifying just the key is equivalent to *key*=`true`; not specifying the key is equivalent to *key*=`false`. Possible keys are listed below:

bb — The bounding box of the graphics image. Its value field must contain four dimensions, separated by spaces.

`bbllx,bblly,bburx,bbury` The lower-left and upper-right x and y coordinates (obsolete[1]).

`hiresbb` Makes LaTeX search for `%%HiResBoundingBox` comments instead of the normal `%%BoundingBox`. Some applications use this key to specify more precise bounding boxes, because the numbers can normally have only integer values. It is a Boolean, either "`true`" or "`false`".

`viewport` Takes four arguments (like `bb`), but in this case the origin is identified with respect to the bounding box specified in the file. To view a 20bp square at the lower-left corner of the picture, for example, you would specify `viewport=0 0 20 20`.

`trim` Similar to the `viewport` key, but the four dimensions correspond to the amount of space to be trimmed (cut off) at the left-hand side, bottom, right-hand side, and top of the included graphics.

`natheight,natwidth` The natural height and width of figure.[2]

`angle` The rotation angle (in degrees, counterclockwise).

`origin` The origin for the rotation, similar to the `origin` parameter of the `\rotatebox` command described on page 632 and in Figure 10.2 on page 632.

`width` The required width (the width of the image is scaled to that value).

`height` The required height (the height of the image is scaled to that value).

`totalheight` The required total height (height + depth of the image is scaled to that value). This key should be used instead of `height` if images are rotated more than 90 degrees, because the height can disappear (and become the `depth`) and LaTeX may have difficulties satisfying the user's request.

`keepaspectratio` A Boolean variable that can have the value "`true`" or "`false`" (see above for defaults). When it is `true`, specifying both the `width` and `height` parameters does not distort the picture, but the image is scaled so that neither the `width` nor `height` *exceeds* the given dimensions.

`scale` The scale factor.

`clip` Clip the graphic to the bounding box. It is a Boolean, either "`true`" or "`false`".

[1] Kept for backward compatibility only. `[bbllx=a, bblly=b, bburx=c, bbury=d]` is equivalent to `[bb = a b c d]`, so the latter form should be used.

[2] These arguments can be used for setting the lower-left coordinate to (`0 0`) and the upper-right coordinate to (`natwidth natheight`) and are thus equivalent to `bb=0 0 w h`, where `w` and `h` are the values specified for these two parameters.

`draft`	Locally switch to draft mode. A Boolean-value key, like `clip`.
`type`	The graphics type; see Section 10.2.5.
`ext`	The file extension of the file containing the image data.
`read`	The file extension of the file "read" by LaTeX to determine the image size, if necessary.
`command`	Any command to be applied to the file.

If the size is given without units for the first eight keys (bb through `trim`), then TeX's "big points" (equal to PostScript points) are assumed.

The first ten keys (bb through `natwidth`) specify the size of the image. This information needs to be given in case TeX cannot read the file, the file contains incorrect size information, or you wish to clip the image to a certain rectangle.

The next seven keys (`angle` through `scale`) have to do with scaling or rotation of the included material. Similar effects can be obtained with the graphics package and the `\includegraphics` command by placing the latter inside the argument of a `\resizebox`, `\rotatebox`, or `\scalebox` command (see the examples in Section 10.2.2 and the in-depth discussion of these commands in Section 10.3).

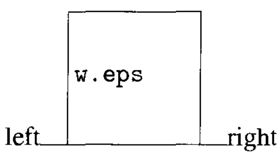
It is important to note that keys are read from left to right, so that `[angle=90, totalheight=2cm]` means rotate by 90 degrees and then scale to a height of 2 cm, whereas `[totalheight=2cm, angle=90]` would result in a final *width* of 2 cm.

By default, LaTeX reserves for the image the space specified either in the file or in the optional arguments. If any part of the image falls outside this area, it will overprint the surrounding text. If the starred form is used or the `clip` option is specified, any part of the image outside this area is not printed.

The last four keys (`type`, `ext`, `read`, `command`) suppress the parsing of the file name. When they are used, the main *file* argument should have no file extension (see the description of the `\DeclareGraphicsRule` command below).

Below we repeat some of the examples from Section 10.2.2 using the syntax of the graphicx package, showing extra facilities offered by the extended package. In most cases the new form is easier to understand than the earlier version. In the simplest case without any optional arguments, the syntax for the `\includegraphics` command is the same in both packages.

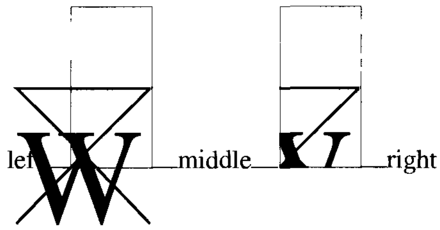
If we use the `draft` key, we get just a frame showing the bounding box. This feature is not offered by the graphics package on the level of individual graphics.




```
\usepackage{graphicx}
% \HR as before
left\HR
 \includegraphics[draft]{w.eps}%
\HR right
```

10-2-6

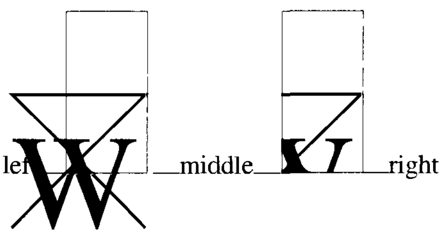
The effects of the bb, `clip`, `viewport`, and `trim` keys are seen in the following examples. Compare them with Example 10-2-2 on page 617.



10-2-7

```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR\bluefbox{\includegraphics
     [bb=120 120 150 180]{w.eps}}%
\HR middle\HR
 \bluefbox{\includegraphics
     [bb=120 120 150 180,clip]{w.eps}}%
\HR right
```

Using `viewport` or `trim` allows us to specify the desired result in yet another way. Notice that we actually trim a negative amount, effectively enlarging the space reserved for the picture.



10-2-8

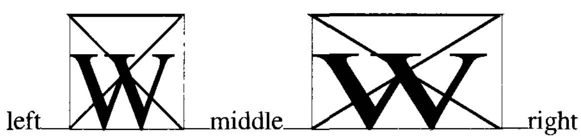
```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR\bluefbox{\includegraphics
     [viewport=20 20 50 80]%
     {w.eps}}
\HR middle\HR
\bluefbox{\includegraphics
     [trim= 20 20 0 -30,clip]{w.eps}}%
\HR right
```

If you want to apply a scale factor to the image, use the `scale` key. With this key, however, you can only scale the picture equally in both directions.

10-2-9

```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR \bluefbox{\includegraphics[scale=.5]{w.eps}}\HR right
```

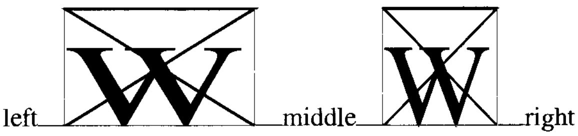
To make the dimensions of an image equal to a given value, use the `width` or `height` key (the other dimension is then scaled accordingly). If you use both keys simultaneously, you can distort the image to fit a specified rectangle, as shown in the following example:



10-2-10

```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR \bluefbox{\includegraphics
    [width=15mm]{w.eps}}%
\HR middle\HR
  \bluefbox{\includegraphics
    [height=15mm,width=25mm]{w.eps}}%
\HR right
```

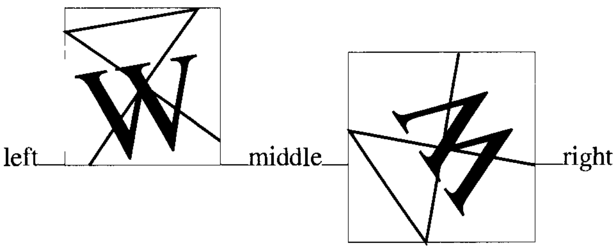
You can make sure that the aspect ratio of the image itself remains intact by specifying the `keepaspectratio` key. LaTeX then fits the image as best it can to the rectangle you specify.




```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR \bluefbox{\includegraphics
      [height=15mm,width=25mm]{w.eps}}%
\HR middle\HR
 \bluefbox{\includegraphics[height=15mm,
   width=25mm,keepaspectratio]{w.eps}}%
\HR right
```

10-2-11

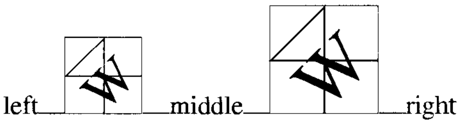
Rotations using the `angle` key add another level of complexity. The reference point for the rotation is the reference point of the original graphic—normally the lower-left corner if the graphic has no depth. By rotating around that point, the height and depth change so that the graphic moves up and down with respect to the baseline, as can be seen in the next examples.




```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR
  \bluefbox{\includegraphics
             [angle=10]{w.eps}}%
\HR middle\HR
  \bluefbox{\includegraphics
             [angle=125]{w.eps}}%
\HR right
```

10-2-12

The real fun starts when you specify both a dimension and a rotation angle for an image, since the order in which they are given matters. The graphicx package interprets the keys *from left to right*. You should pay special attention if you plan to rotate images and want to set them to a certain height. The next examples show the difference between specifying an angle of rotation before and after a scale command. In the first case, the picture is rotated and then the result is scaled. In the second case, the picture is scaled and then rotated.

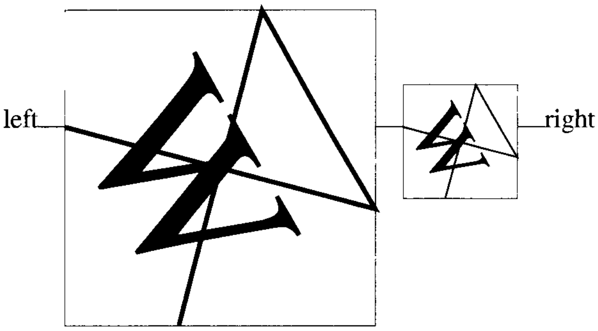



```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR\bluefbox{\includegraphics
      [angle=45,width=10mm]{w.eps}}%
\HR middle\HR
\bluefbox{\includegraphics
      [width=10mm,angle=45]{w.eps}}%
\HR right
```

10-2-13

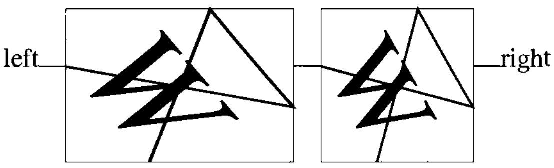
LaTeX considers the height and the depth of the rotated bounding box separately. The `height` key refers only to the height; that is, it does not include the depth. In general, the total height of a (rotated) image should fit in a given space, so you should use the `totalheight` key (see Figure 10.2 on page 632 for a description of the various dimensions defining a LaTeX box). Of course, to obtain special effects you can manipulate rotations and combinations of the `height` and `width` parameters at will. Here we show some key combinations and their results.

10-2-14




```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR\bluefbox{%
  \includegraphics[angle=-60,%
                   height=15mm]%
    {w.eps}}\HR
\bluefbox{%
  \includegraphics[angle=-60,%
                   totalheight=15mm]%
    {w.eps}}\HR right
```

10-2-15




```
\usepackage{graphicx,color}
% \bluefbox and \HR as before
left\HR\bluefbox{\includegraphics
      [angle=-60,totalheight=20mm,%
       width=30mm]{w.eps}}\HR
\bluefbox{\includegraphics
      [angle=-60,totalheight=20mm,%
       width=30mm,keepaspectratio]%
      {w.eps}}\HR right
```

10.2.4 Setting default key values for the graphicx package

Instead of specifying the same set of key/value pairs over and over again on individual `\includegraphics` commands, you can specify global default values for keys associated with such commands. To do so, you use the `\setkeys` declaration provided by the keyval package, which is automatically included when graphicx is used.

`\setkeys{`*identifier*`}{`*key/val-list*`}`

The *identifier* is an arbitrary string defined by the macro designer. For example, for `\includegraphics` the string `Gin` was chosen. The *key/val-list* is a comma-separated list of key/value pairs.

As an example, consider the case where graphicx is used and all figures are to be scaled to the width of the line. Then you would specify the following:

```
\setkeys{Gin}{width=\linewidth}
```

All images included with the `\includegraphics` command will then be automatically scaled to the current line width. (Using `\linewidth` in such a case is usually preferable to using `\columnwidth`, as the former changes its value depending on the surrounding environment, such as `quote`.)

You can specify defaults in a similar way for any key used with the `\rotatebox` command (the other command that has a key/value syntax when graphicx is used). It has the *identifier* `Grot`; thus,

```
\setkeys{Grot}{origin=ct}
```

specifies that `ct` should be used for the `origin` key on all `\rotatebox` commands unless locally overwritten.

10.2.5 Declarations guiding the inclusion of images

While key/value pairs can be set only when the graphicx package is used, the declarations described in this section can be used with both the graphics and the graphicx packages.

Where to find image files

By default, LaTeX looks for graphics files in the same directories where it looks for other files. But for larger projects it might be preferable to keep the image files together in a single directory or in a set of directories. A list of directories where LaTeX should search for graphics files can be specified through the command `\graphicspath`, whose argument is a list of directories, each inside a pair of braces `{}` (even if the list contains only one directory). For example,

```
\graphicspath{{./eps/}{./tiff/}}
```

causes LaTeX to look in the subdirectories `eps` and `tiff` of the current directory.

Defining the file extension search order

The `\DeclareGraphicsExtensions` command lets you specify the behavior of the system when no file extension is given in the argument of the `\includegraphics` command. Its argument {*ext-list*} is a comma-separated list of file extensions. Full file names are constructed by appending each extension of the list *ext-list* in turn until a file corresponding to the generated full file name is found.

Because the algorithm tests for the existence of a file to determine which extension to use, when the `\includegraphics` command is specified without an extension, the graphics file must exist at the time LaTeX is run. However, if a file extension *is* specified, such as `\includegraphics{gr.eps}` instead of `\includegraphics{gr}`, then the graphics file need not exist at the time of the

LaTeX run.[1] LaTeX needs to know the size of the image, however, so it must be specified in the arguments of the `\includegraphics` command or in a file actually read by LaTeX. (This file can be either the graphics file itself or another file specified with the `read` key or constructed from the list of file extensions. In the latter case the file must exist at the time LaTeX is run.)

With the declaration shown below, the `\includegraphics` command will first look for the file `file.ps` and, if no such file exists, for the file `file.ps.gz`:

```
\DeclareGraphicsExtensions{.ps,.ps.gz}
  \includegraphics{file}
```

If you want to make sure that a full file name must always be specified, then you should use the following declaration. In the cases shown below, the size of the (bitmap) image is specified explicitly on the `\includegraphics` command each time.

```
\DeclareGraphicsExtensions{{}}
  \includegraphics[1in,1in]{file.pcx}
  \includegraphics[75pt,545pt][50pt,530pt]{file.pcx}
  \includegraphics[bb=75 545 50 530]{file.pcx}
```

The action that has to take place when a file with a given extension is encountered is controlled by the following command:

`\DeclareGraphicsRule{`*ext*`}{`*type*`}{`*read-file*`}{`*cmd*`}`

Any number of these declarations is allowed. The meanings of the arguments are described below.

ext The extension of the image file. It can be specified explicitly or, if the argument to `\includegraphics` does not have an extension, can be determined from the list of extensions specified in the argument *ext-list* of the `\DeclareGraphicsExtensions` command. A star (`*`) can be used to specify the default behavior for all extensions that are not explicitly declared. For example,

```
\DeclareGraphicsRule{*}{eps}{*}{}
```

causes all undeclared extensions to be treated as EPS files, and the respective graphics files are read to search for a `%%BoundingBox` comment.

type The "type" of the file involved. All files of the same type are input with the same internal command (which must be defined in the corresponding driver file). For example, files with an extension of `.ps`, `.eps`, or `.ps.gz` should all be classified as being of type `eps`.

[1]For instance, it can be created on the fly with a suitable `\DeclareGraphicsRule` declaration.

	ext	*type*	*read-file*	*cmd*
Basic PostScript	`.ps`	eps	`.ps`	
	`.eps`	eps	`.eps`	
Dynamic Decompression	`.pz`	eps	`.bb`	`` `gunzip -c #1``
	`.ps.gz`	eps	`.ps.bb`	`` `gunzip -c #1``
	`.eps.gz`	eps	`.eps.bb`	`` `gunzip -c #1``
MS-DOS-related Formats	`.tif`	tiff		
	`.pcx`	bmp		
	`.bmp`	bmp		
	`.msp`	bmp		
Mac-related Formats	`.pict`	pict		
	`.pntg`	pntg		

Table 10.2: Arguments of `\DeclareGraphicsRule`

read-file The extension of the file that should be read to determine the size of the graphics image. It can be identical to *ext*, but, in the case of compressed or binary images, which cannot be interpreted easily by LaTeX, the size information (the bounding box) is normally put in a separate file. For example, for compressed gzipped PostScript files characterized by the extension `.ps.gz`, the corresponding readable files could have extension `.ps.bb`. If the *read-file* argument is empty (i.e., `{}`), then the system does not look for an external file to determine the size, and the size must be specified in the arguments of the command `\includegraphics`. If the driver file specifies a procedure for reading size files for *type*, then that procedure is used; otherwise, the procedure for reading `.eps` files is used. Therefore, in the absence of any other specific format, you can select the size of a bitmap picture by using the syntax for PostScript images (i.e., with a `%%BoundingBox` line).

cmd The command to be inserted in the `\special` argument instead of the file name. In general *cmd* is empty, but for compressed files you might want to uncompress the image file before including it in the file to be printed if the driver supports such an operation. For instance, with the dvips driver, you could use

```
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{`gunzip #1}
```

where the argument `#1` denotes the full file name. In this case the final argument causes dvips to use the gunzip command to uncompress the file before inserting it into the PostScript output.

Various possibilities for the arguments of the `\DeclareGraphicsRule` command are shown in Table 10.2.

The system described so far can give some problems if the extension *ext* does not correspond to the *type* argument. One could, for instance, have a series of PostScript files called `file.1`, `file.2`, ... Neither the graphics nor the graphicx package can automatically detect that these are PostScript files. With the graphicx package, this determination can be handled by using a `type=eps` key setting on each `\includegraphics` command. To handle this situation more generally, you can define a default type by using a `\DeclareGraphicsRule` declaration for a type `*` as explained above.

10.2.6 A caveat: Encapsulation is important

We will describe PostScript in more detail in Section 10.4, but it is already important at this point to emphasize that PostScript is a page description language that deals with the appearance of a *complete printed page.* This makes it difficult for authors to include smaller PostScript pictures created by external tools into their electronic (LaTeX) documents. To solve this problem Adobe has defined the *Encapsulated PostScript* file format (EPS or EPSF), which complies with the *PostScript Document Structuring Conventions Specification* [2] and the *Encapsulated PostScript File Format Specification* [3].

The EPS format defines standard rules for importing PostScript language files into different environments. In particular, so as not to interfere destructively with the PostScript page being built, EPS files should be "well behaved". For instance, they must not contain certain PostScript operators, such as those manipulating the graphics state, interpreter stack, and global dictionaries.

Most modern graphics applications generate an EPS-compliant file that can be used without difficulty by LaTeX. Sometimes, however, you may be confronted with a bare PostScript file that does not contain the necessary information. For use with LaTeX, a PostScript file does not have to conform strictly to the structuring conventions mentioned previously. If the file is "well behaved" (see above), it is enough that the PostScript file contains the dimensions of the box occupied by the picture. These dimensions are provided to LaTeX via the PostScript comment line `%%BoundingBox`, as shown below:

```
%!
%%BoundingBox: LLx LLy URx URy
```

The first line indicates that we are dealing with a nonconforming EPS file. Note that the `%!` characters *must* occupy the first two columns of the line. The second line, which is the more important one for our purpose, specifies the size of the included picture in PostScript "big" points, of which there are 72 to an inch (see Table A.1 on page 855). Its four parameters are the x and y coordinates of the lower-left corner (`LLx` and `LLy`) and the upper-right corner (`URx` and `URy`) of the picture. For instance, a full A4 page (210 mm by 297 mm) with zero at the lower-left corner would need the following declaration:

```
%!
%%BoundingBox: 0 0 595 842
```

If your picture starts at (100, 200) and is enclosed in a square of 4 inches (288 points), the statement would be

```
%!
%%BoundingBox: 100 200 388 488
```

A PostScript display program, such as ghostview, lets you easily determine the bounding box of a picture by moving the cursor on its extremities and reading off the corresponding coordinates. In general, it is good practice to add one or two points to make sure that the complete picture will be included, because of the potential for rounding errors during the computations done in the interpreter.

10.3 Manipulating graphical objects in LaTeX

In addition to the `\includegraphics` command, the graphics and graphicx packages implement a number of graphical manipulation commands.

With the exception of the `\rotatebox` command, which also supports a key/value pair syntax in the graphicx package, the syntax for these commands is identical in both packages.

10.3.1 Scaling a LaTeX box

The `\scalebox` command lets you magnify or reduce text or other LaTeX material by a scale factor.

`\scalebox{`*h-scale*`}[`*v-scale*`]{`*material*`}`

The first of its arguments specifies the factor by which both dimensions of the *material* are to be scaled. The following example shows how this works.

This text is normal.
This text is large.
This text is tiny

```
\usepackage{graphics} % or graphicx
\noindent This text is normal.  \\
\scalebox{2}{This text is large.}\\
\scalebox{0.5}{This text is tiny.}
```
10-3-

A supplementary optional argument, if present, specifies a separate vertical scaling factor. It is demonstrated in the following examples, which also show how multiple lines can be scaled by using the standard LaTeX \parbox command.

10-3-2

America & Europe | America & Europe

```
\usepackage{graphics} % or graphicx
\fbox{\scalebox{1.5}{%
 \parbox{.5in}{America \&\\Europe}}}
\fbox{\scalebox{1.5}[1]{%
 \parbox{.5in}{America \&\\Europe}}}
```

`\reflectbox{`*material*`}`

This command is a convenient abbreviation for `\scalebox{-1}[1]{`*material*`}`, as seen in the following example:

10-3-3

America?America?
America?America?

```
\usepackage{graphics} % or graphicx
\noindent America?\reflectbox{America?}   \\
          America?\scalebox{-1}[1]{America?}
```

More interesting special effects can also be obtained. Note in particular the use of the zero-width \makebox commands, which hide their contents from LaTeX and thus offer the possibility of fine-tuning the positioning of the typeset material.

10-3-4

America?America?
America?America?
America?America?
America?America?

```
\usepackage{graphics} % or graphicx
\noindent America?\scalebox{-1}{America?}   \\
          America?\scalebox{1}[-1]{America?}\\
          America?\makebox[0mm][r]{%
                    \scalebox{-1}{America?}}\\
          \makebox[0mm][l]{America?}%
                   \scalebox{1}[-1]{America?}
```

10.3.2 Resizing to a given size

It is possible to specify that LaTeX material should be typeset to a fixed horizontal or vertical dimension:

`\resizebox*{`*h-dim*`}{`*v-dim*`}{`*material*`}`

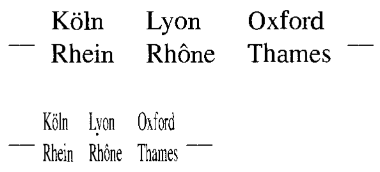
When the aspect ratio of the material should be maintained, then it is enough to specify one of the dimensions, replacing the other dimension with a "!" sign.

10-3-5

London, Berlin & Paris | London, Berlin & Paris

```
\usepackage{graphics} % or graphicx
\fbox{\resizebox{5mm}{!}{%
  \parbox{14mm}{London,\\ Berlin \&\\ Paris}}}
\fbox{\resizebox{!}{10mm}{%
  \parbox{14mm}{London,\\ Berlin \&\\ Paris}}}
```

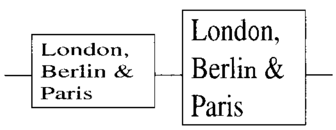
When explicit dimensions for both *h-dim* and *v-dim* are supplied, then the contents can be distorted. In the following example the baseline is indicated by a horizontal rule drawn with the \HR command.



```
\usepackage{graphics} % or graphicx
\HR\begin{tabular}{lll}
      K\"oln & Lyon    & Oxford \\
      Rhein  & Rh\^one & Thames
 \end{tabular}\HR\par\bigskip
\HR\resizebox{2cm}{.5cm}{%
    \begin{tabular}{lll}
      K\"oln & Lyon    & Oxford \\
      Rhein  & Rh\^one & Thames
    \end{tabular}}\HR
```

10-3-6

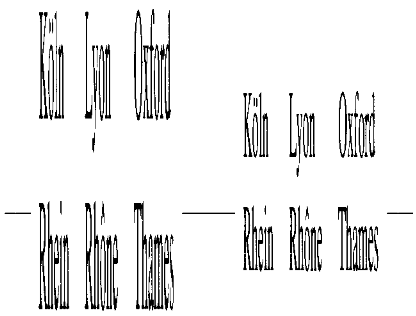
As usual with LaTeX commands involving box dimensions, you can refer to the natural lengths \depth, \height, \totalheight, and \width as dimensional parameters:



```
\usepackage{graphics} % or graphicx
\HR\fbox{\resizebox{\width}{.7\height}{%
   \parbox{14mm}{London,\\ Berlin \&\\Paris}}}\HR
\fbox{\resizebox{\width}{.7\totalheight}{%
  \parbox{14mm}{London,\\ Berlin \&\\Paris}}}\HR
```

10-3-7

The unstarred form \resizebox bases its calculations on the height of the LaTeX material, while the starred \resizebox* command takes into account the total height (the depth plus the height) of the LaTeX box. The next tabular examples, which have a large depth, show the difference.



```
\usepackage{graphicx}
\HR\resizebox{20mm}{30mm}{%
     \begin{tabular}{lll}
       K\"oln & Lyon    & Oxford \\
       Rhein  & Rh\^one & Thames
     \end{tabular}}\HR
\HR\resizebox*{20mm}{30mm}{%
     \begin{tabular}{lll}
       K\"oln & Lyon    & Oxford \\
       Rhein  & Rh\^one & Thames
     \end{tabular}}\HR
```

10-3-8

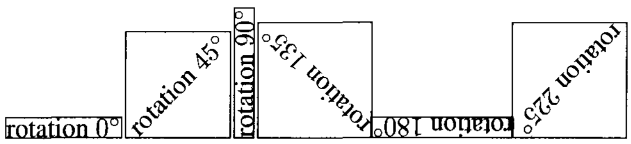
10.3.3 Rotating a LaTeX box

LaTeX material can be rotated through an angle with the \rotatebox command. An alternative technique useful with environments is described in Section 10.3.4.

`\rotatebox{`*angle*`}{`*material*`}`

The *material* argument is typeset inside a LaTeX box and rotated through *angle* degrees counterclockwise around the reference point.

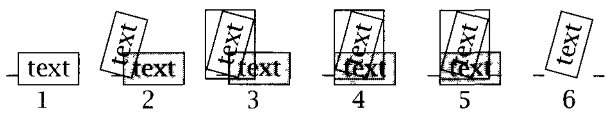
10-3-9




```
\usepackage{graphics} % or graphicx
\newcommand\MyRot[1]{\frame
   {\rotatebox{#1}{rotation
                          $#1^\circ$}}}
\MyRot{0}  \MyRot{45} \MyRot{90}
\MyRot{135}\MyRot{180}\MyRot{225}
```

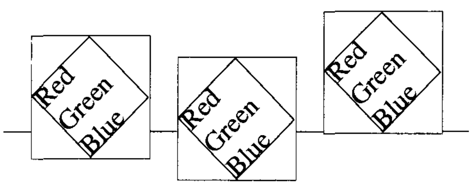
The rotation algorithm

To understand where the rotated material is placed on the page, we need to look at the algorithm employed. Below we show the individual steps carried out when rotating `\fbox{text}` by 75 degrees. Step 1 shows the unrotated text; the horizontal line at the left marks the baseline. First the *material* (in this case, `\fbox{text}`) is placed into a box. This box has a reference point around which, by default, the rotation is carried out. This point is shown in step 2 (the original position of the unrotated material is shown as well for reference purposes). Then the algorithm calculates a new bounding box (i.e., the space reserved for the rotated material), as shown in step 3. Next the material is moved horizontally so that the left edges of the new and the old bounding boxes are in the same position (step 4). TeX's typesetting position is then advanced so that additional material is typeset to the right of the bounding box in its new position, as shown by the line denoting the baseline in step 5. Step 6 shows the final result, again with the baseline on both sides of the rotated material.



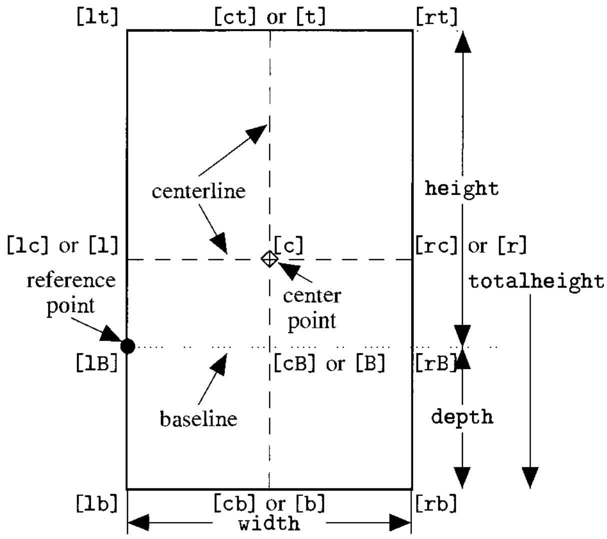
For more complex material it is important to keep in mind the location of the reference point of the resulting box. The following example shows how it can be shifted by using the placement parameter of the `\parbox` command.

10-3-10




```
\usepackage{color,graphics} % or graphicx
\HR\bluefbox{\rotatebox{45}{%
   \fbox{\parbox{3em}{Red\\Green\\Blue}}}}%
\HR\bluefbox{\rotatebox{45}{%
   \fbox{\parbox[t]{3em}{Red\\Green\\Blue}}}}%
\HR\bluefbox{\rotatebox{45}{%
   \fbox{\parbox[b]{3em}{Red\\Green\\Blue}}}}\HR
```

The extended graphics package graphicx offers more flexibility in specifying the point around which the rotation is to take place by using *key/val* pairs.



Horizontal alignment	`l`	left	`r`	right	`c`	center
Vertical alignment	`t`	top	`b`	bottom	`B`	baseline

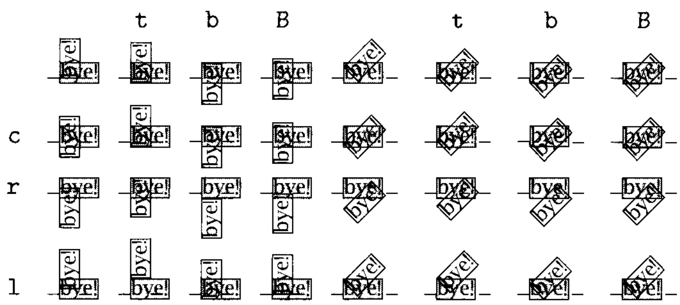
10-3-11

Figure 10.2: A LaTeX box and possible `origin` reference points

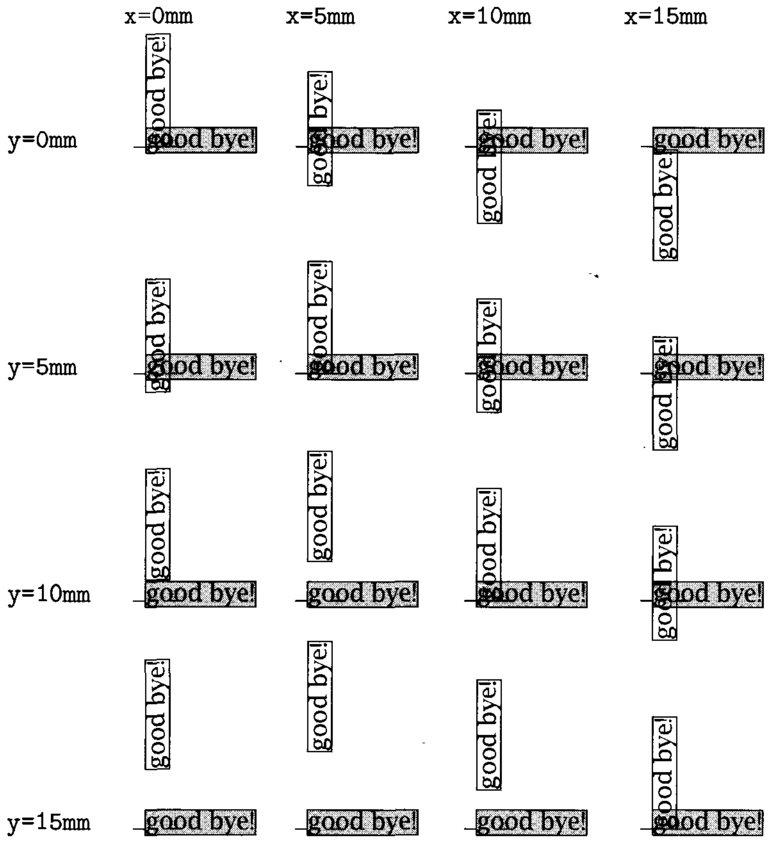
`\rotatebox`[*key/val-list*]{*angle*}{*material*}

The four possible keys in this case are `origin`, `x`, `y`, and `units`. The possible values for the `origin` key are shown in Figure 10.2 (one value each for the horizontal and vertical alignments can be chosen), as are the actual positions of these combinations with respect to the LaTeX box produced from *material*.

The effect of these possible combinations for the `origin` key on an actual LaTeX box can be studied below, where two matrices of the results are shown for 90-degree and 45-degree rotated boxes. To better appreciate the effects, the unrotated text is shown against a grey background.



If the specification of the `origin` is not enough, you also can supply the x and y coordinates (relative to the reference point) for the point around which the rotation is to take place. For this purpose, use the keys `x` and `y` and the format `x=`*dim*`,` `y=`*dim*. A matrix showing some sample values and their effect on a box rotated by 90 degrees appear below.



The interpretation of the *angle* argument of `\rotatebox` can be controlled by the `units` keyword, which specifies the number of units counterclockwise in a full circle. The default is 360, so using `units=-360` would mean that angles are specified clockwise. Similarly, a setting of `units=6.283185` changes the degree specification to radians. Rather than changing the `units` key on individual `\rotatebox` commands, you should probably set up a default interpretation using the `\setkeys` declaration as described in Section 10.2.4.

10.3.4 rotating—Revisited

The material in this section is similar to that of Sebastian Rahtz's rotating package, which was introduced in Section 6.3.3 on page 296. The functionality of rotating is implemented in this package through the environments `turn` and `rotate`; the latter environment generates an object that occupies no space. Using environments has the advantage that the rotated material can contain `\verb` commands. However, the extended syntax of the `\rotatebox` command is not supported, so in most cases the latter command is preferable.

Turning LaTeX \LaTeX a bit.

```
\usepackage{rotating}
Turning \begin{rotate}{-20}\Large\LaTeX\end{rotate}%
        \begin{turn}{20}\verb=\LaTeX=\end{turn} a bit.
```

10-3-12

10.4 Display languages: PostScript, PDF, and SVG

After typesetting an electronic document, one usually would like to view the generated output "page"—on paper via a printing device, on a PC screen, with a dedicated program or inside a browser, or (why not?) on a portable phone.

Several display languages have been developed over the years. For printing devices PostScript, which is essentially a language for describing a static output page, has become the most important player. In the early 1990s, Adobe developed a light-weight version of PostScript, called the Portable Document Format (PDF) [5]. PDF implements a similar imaging model as PostScript but introduces a more structured format to improve performance for interactive viewing. It also adds links and annotations for navigation.

The increasing affordability of the personal computer has drastically reduced the production cost of electronic documents. The World Wide Web makes distributing these documents worldwide cheap, easy, and fast. The development of the XML family of standards has made it possible to apply a unified approach to handle the huge amount of information stored electronically and to transform it into various customizable presentation forms.

Various techniques are now available to transform LaTeX documents into PDF, HTML (XHTML), or XML so that the information can be made available on the web (several chapters of *The Web Companion* [56] are dedicated to explaining such techniques). A particularly interesting approach, described below, involves transforming LaTeX-encoded information into a Scalable Vector Graphics (SVG) format.

Thus, LaTeX can continue to play a major role in the integrated worldwide cyberspace. Especially in the area of scientific documents, it will remain an important (intermediate) format for generating high-quality printable PDF or browsable SVG output.

This section gives a short introduction to these three display languages—PostScript, PDF, and SVG. It briefly describes dvips, a `dvi`-to-PostScript translator, and discusses pspicture, an enhancement of LaTeX's `picture` environment using PostScript.

10.4.1 The PostScript language

PostScript [4] is a page description language. It provides a method for expressing the appearance of a printed page, including text, lines, and graphics.

A device- and resolution-independent, general-purpose, programming language, PostScript describes a complete "output page". The language is stack oriented and uses "reverse Polish" or postfix notation. It includes looping constructs, procedures, and comparison operators, and it supports many data types, including reals, Booleans, arrays, strings, and complex objects such as dictionaries.

PostScript programs are generally written in the form of ASCII source text, which is easy to create, understand, transmit, and manipulate. Because PostScript is resolution and device independent, the same ASCII file can be viewed on a computer display with a previewer, such as ghostscript/ghostview, and printed on a small laser printer or a high-resolution phototypesetter.

The PostScript language lets you mix the following features in any number of combinations:

- Arbitrary shapes can be constructed from lines, arcs, and cubic curves. The shapes may self-intersect and contain disconnected sections and holes.
- The painting primitives permit shapes to be outlined with lines of any thickness, filled with any color, or used as a clipping path to crop any other graphic.
- Text is fully integrated with graphics. In PostScript, text characters are treated as graphical shapes that may be operated on by any of the language's graphics operators. This is fully true for Type 3 fonts, where character shapes are defined as ordinary PostScript language procedures. In contrast, Adobe's Type 1 format defines a special smaller language where character shapes are defined by using specially encoded procedures (see below). For complex languages with many thousands of characters (e.g., Chinese and Japanese), composite Type 0 fonts can be used.
- Images (such as photographs or synthetically generated images) can be sampled at any resolution and with a variety of dynamic ranges. PostScript provides facilities to control the rendering of images on the output device.
- Several color models (device based: RGB, HSB, CMYK; standard based: CIE) are available, and conversion from one model to another is possible.
- A general coordinate system facility supports all combinations of linear transformations, including scaling, rotation, reflection, and skewing. These transformations apply uniformly to all page elements, including text, graphical images, and sampled images.
- Dictionaries for color spaces, fonts, forms, images, half-tones, and patterns are available.
- Compression filters, such as JPEG and LZW, are available.

Type 1 and OpenType font outlines

As a complement to the PostScript language, Adobe has defined its Type 1 font format [1]. A Type 1 font program consists of a clear text (ASCII) portion, plus an encoded and encrypted portion. The PostScript language commands used in a Type 1 font program conform to a much stricter syntax than do normal PostScript language programs.

Adobe's Type 1 model is, like PostScript, fully device and resolution independent. It uses mathematical expressions—in particular, Bézier curves—to define character outlines, thereby guaranteeing flexibility and rendering accuracy. Characters are defined at a size of 1 point in a 1000 by 1000 coordinate system, which can then be scaled, rotated, and skewed at will. Hints can be included to make the representation as exact as possible on a wide variety of devices and pixel densities.

Recently, Adobe and Microsoft jointly developed OpenType,[1] a new cross-platform font file format. This extension of the TrueType font outline format can also support Type 1 font data. OpenType adds new typographic features as well.

You can move OpenType font files back and forth between platforms (Macintosh and Windows), improving cross-platform portability for any documents that use these types. The bitmap, outline, and metric data are combined into a single, cross-platform OpenType font file, simplifying font management.

OpenType fonts are based on Unicode, an international multi-byte character encoding that covers virtually all of the world's languages. OpenType thus makes multilingual typography easier by including multiple language character sets in one font. The basic OpenType fonts contain the standard range of Latin characters used in the Western world, as well as several international characters (e.g., the euro symbol). Pro versions add a full range of accented characters to support Central and Eastern European languages, such as Turkish and Polish, and many contain Cyrillic and Greek character extensions in the same font.

Given that OpenType fonts may contain more than 65,000 glyphs, they provide far more typographic capabilities by combining base character sets, expert sets, and extensive additional glyphs into one file. For instance, a single font file may contain many nonstandard glyphs, such as old-style figures, true small capitals, fractions, swashes, superiors, inferiors, titling letters, contextual and stylistic alternates, and a full range of ligatures.

OpenType manages the mapping between characters and glyphs. In particular, its layout features can be used to position or substitute glyphs. For any character, there is a default glyph and positioning behavior. The application of layout features to one or more characters may change the positioning, or substitute a different glyph.

Over the years, thousands of typefaces, including those of the world's major typesetting companies, such as Linotype, Agfa-Compugraphic, Monotype, Autologic, and Varityper, have become available in PostScript Type 1 format. More

[1] See `http://partners.adobe.com/asn/developer/opentype/main.html`.

recently, Adobe has converted the entire Adobe Type Library (thousands of fonts) into OpenType, and other type foundries are following Adobe's example.

In the TeX world, the Ω (Omega) program (`http://omega.cse.unsw.edu.au`), an extension of TeX developed by Yannis Haralambous and John Plaice that features multi-byte data structures and is based on Unicode for its internal character representation, can take advantage of OpenType fonts.

10.4.2 The dvips PostScript driver

Tom Rokicki's dvips program[1] is undoubtedly the most widely used `dvi`-to-PostScript driver. It is a very mature product, with many important and useful features. The `\special` support in dvips is extensive; in particular, it supports the pic commands of the eepic package mentioned in Section 10.1.5.

The dvips program will automatically generate missing fonts if METAFONT exists on the system. If a font cannot be generated, a scaled version of the same font at a different size will be used instead (although dvips will complain about the poor aesthetics of the resulting output). Moreover, this facility is configurable and is not limited simply to running METAFONT.

The output from dvips can be controlled in two ways: by command-line switches for a particular job and by commands in one or more configuration files. Using configuration files, you can set parameters globally for the whole system, on a per-printer basis, and on a per-user basis.

When dvips starts up, a global `config.ps` file is searched for.[2]

The dvips driver has a plethora of command-line options. Table 10.3 on the following page presents a summary of those options.

With the help of the `-d` option for dvips, you can track down errors and understand what is going on. You must supply an integer specifying the class of information to be displayed. To get several types of information, simply add the numbers together for the types in which you are interested. Choose from the following:

1	specials	4	fonts	16	headers	64	files
2	paths	8	pages	32	font compression	128	memory

For example, calling dvips with the `-d 4` option yields information about which fonts are being called and where they are loaded from. An option of `-d -1` (all flags are activated) displays a very detailed log of everything dvips does. It will, however, generate an enormous volume of data, so this facility should be used only as a last resort, if a more refined approach fails.

[1] The manual is at `http://www.ctan.org/tex-archive/dviware/dvips/dvips_man.pdf`. See also [57, Chapter 11] for a detailed description.

[2] This file must exist on the search path of dvips which is usually something like `texmf/dvips/config` below the root of the TeX installation tree.

`a*`	Conserve memory, not time	`y #`	Multiply by dvi magnification
`b #`	Page copies, e.g., for posters	`z*`	Hyper PostScript
`c #`	Uncollated copies	`A`	Print only odd (TEX) pages
`d #`	Debugging	`B`	Print only even (TEX) pages
`e #`	Maxdrift value	`C #`	Collated copies
`f*`	Run as filter	`D #`	Resolution
`h f`	Add header file	`E*`	Try to create EPSF
`i*`	Separate file per section	`F*`	Send `control-D` at end
`k*`	Print crop marks	`G*`	Shift low chars to higher pos.
`l #`	Last page	`K*`	Pull comments from inclusions
`m*`	Manual feed	`M*`	Don't make fonts
`n #`	Maximum number of pages	`N*`	No structured comments
`o f`	Output file	`O c`	Set/change paper offset
`p #`	First page (p=# absolute)	`P s`	Load `config.$s`
`pp#`	One page only	`R`	Run securely
	ppn_1:n_2 page range	`S #`	Max section size in pages
`q*`	Run quietly	`T c`	Specify desired page size
`r*`	Reverse order of pages	`U*`	Disable string param trick
`s*`	Enclose output in save/restore	`X #`	Horizontal resolution
`t s`	Paper format	`Y #`	Vertical resolution
`x #`	Override dvi magnification	`Z*`	Compress bitmap fonts

= number   `f` = file name   `s` = string   `*` = suffix, '`0`' to turn off
`c` = comma-separated dimension pair (e.g., `3.2in`, `-32.1cm`)

Table 10.3: Major options of the dvips program

10.4.3 pspicture—An enhanced `picture` environment for dvips

David Carlisle's pspicture package reimplements, and extends, LaTeX's `picture` environment with the help of PostScript commands that are placed in TeX `\special` commands. It eliminates limitations in standard LaTeX where `picture` offers only a discrete range of slopes and thicknesses for lines and a limited range of diameters for circles.

There exists a certain amount of overlap between this package and the eepic package, described earlier. Moreover, the pspicture package can be considered as a sort of "stand-in" for the pict2e package that was announced by Leslie Lamport in 1994 in the second edition of the LaTeX book, but which was never written.[1]

However, pspicture has the disadvantage that a picture can no longer be

[1]For the next LaTeX release a first implementation of the pict2e package (by Hubert Gaßlein and Rolf Niepraschk) is being considered for inclusion in LaTeX.

viewed with a dvi program that has no facility to interpret and display PostScript commands.[1] A "poor man's" workaround is the companion package texpicture. It uses the standard picture commands as much as possible, but silently omits any picture object that cannot be drawn with standard LaTeX. Of course, the visual result in this case will probably not conform to the finally envisaged version—but at least the document will compile.

The dvi file produced with pspicture contains embedded \special commands that are set up to be recognized by Rokicki's dvips driver. Thus, the driver file pspicture.ps, which contains the PostScript code referenced in the \special commands for use by the downstream PostScript interpreter, must be present on the TeX installation in the relevant dvips directory, so that it can be found and included by dvips when needed.

Extended or changed commands

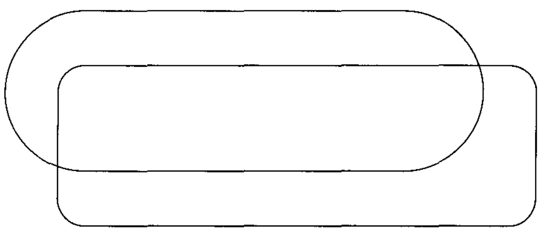
The pspicture package extends the functionality of several commands that are available inside LaTeX's picture environment.

The \circle and \circle* commands are similar to their counterparts in standard LaTeX but have no limit on their diameters. The thickness of the circle is altered by the \linethickness command. The size of the circle produced by \circle* is not affected by \linethickness. \circle *extensions*

\oval[*radius*] (*x,y*) [*part*]

The \oval command acts as described in the LaTeX book, but there is no maximum diameter for the circular arcs, so the oval (in the absence of the optional parameter [*part*]) always consists of two semicircular arcs joined by a pair of parallel lines. To obtain a "rectangle with rounded corners", a second optional argument *radius* was added at the beginning of the \oval command. If this option is used, \oval works with circular arcs of radius min(*radius*, $x/2, y/2$). The following example shows the difference. \oval *extensions*

10-4-1



```
\usepackage{pspicture}
\begin{picture}(200,120)
  \put(90,40) {\oval     (180,60)}
  \put(110,20){\oval[10](180,60)}
\end{picture}
```

The \vector and \line commands are as described in the LaTeX book but no longer have any restrictions on their slopes. The thickness of a sloping line is altered by the \linethickness command. The arrowheads drawn by the vector \line *and* \vector *extensions*

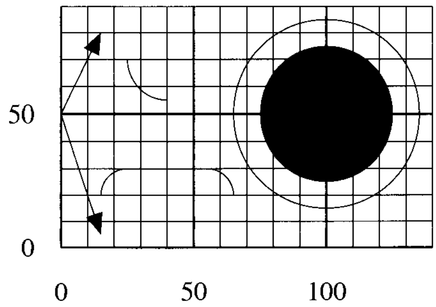
[1] If you use pdftex to generate PDF directly, you will encounter the same problem. In this case pspicture should not be used.

command are of triangular shape, and by default, are larger than LaTeX's defaults. The size can be controlled with the `\arrowlength` command described below.

The `\thinlines`, `\thicklines`, and `\linethickness` commands alter the thickness of *all* lines, including slanted lines and circular arcs.

All other commands of LaTeX's `picture` environment, such as `\dashbox`, `\framebox`, `\makebox`, `\multiput`, `\put`, and `\shortstack`, are unaltered and act as described in the LaTeX book.

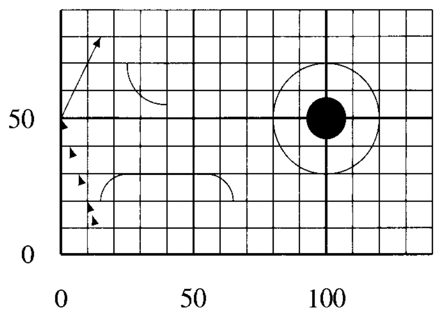
The next example shows how the pspicture package uses PostScript to extend LaTeX's `picture` environment. To allow a better understanding of what is going on, we also use the graphpap's `\graphpaper` command to draw a coordinate grid at a specified position with a given range (first line in the `picture` environment). Here is what pspicture produces.




```
\usepackage{pspicture}\usepackage{graphpap}
\begin{picture}(140,90)
\graphpaper(0,0)(140,90)
\put(0,50){\vector(1,2){15}}
\put(0,50){\vector(2,-6){15}}
\put(40,20){\oval(50,20)[t]}
\put(40,70){\oval(30,30)[bl]}
\put(100,50){\circle{70}}
\put(100,50){\circle*{50}}
\end{picture}
```

10-4-2

To clearly see the effects of the extensions implemented by pspicture, we would like to compare how LaTeX's standard `picture` environment would display the above code. However, these commands cannot be run with LaTeX's `picture` environment, because we have used unsupported arguments for the `\vector`, `\circle`, and `\circle*` commands. Therefore, we must specify the texpicture package instead of pspicture, as shown below. Thanks to the overlayed coordinate grid, the limitations with respect to the pspicture case are clearly visible. Indeed, the second `\vector` is not rendered correctly, while the diameters of the two circles no longer correspond to what is required.




```
\usepackage{texpicture}\usepackage{graphpap}
\begin{picture}(140,90)
\graphpaper(0,0)(140,90)
\put(0,50){\vector(1,2){15}}
\put(0,50){\vector(2,-6){15}}
\put(40,20){\oval(50,20)[t]}
\put(40,70){\oval(30,30)[bl]}
\put(100,50){\circle{70}}
\put(100,50){\circle*{50}}
\end{picture}
```

10-4-3

New commands

The pspicture package also introduces a set of new commands. The `\Line` and `\Vector` commands make it easier to draw a line by allowing you to specify "relative coordinates".

`\put(`x_1,y_1`){\Line(`x_2,y_2`)}` `\put(`x_1,y_1`){\Vector(`x_2,y_2`)}`

The above syntax will result in drawing a line (or a vector) between points (x_1, y_1) and $(x_1 + x_2, y_1 + y_2)$.

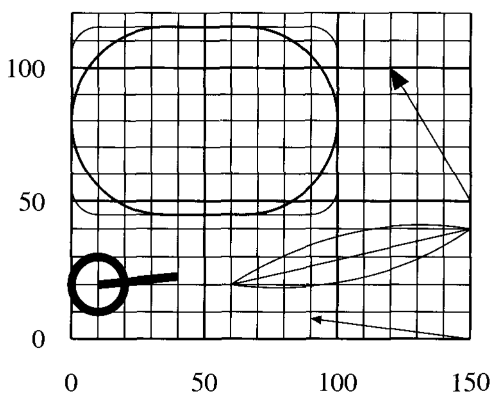
`\put(`x_1,y_1`){\Curve(`x_2,y_2`){`m`}}`

The `\Curve` commands is similar to `\Line`, but generates a line whose curvature is controlled by m (try 1 or -1 first). The value of m does not have to be an integer. Negative numbers curve the line in the opposite way to positive numbers.

`\arrowlength{`*size*`}`

The `\arrowlength` command specifies the size of the triangular arrowhead drawn by the `\vector` and `\Vector` commands. Like `\linethickness`, it is an absolute value (i.e., not affected by `\unitlength`), given in any of LaTeX's units.

Some of the extra features that are not available with the `picture` environment in standard LaTeX are shown below. The possibilities of arbitrary slopes for the `\line` and `\vector` commands were mentioned previously. The more friendly user interface (allowing for relative coordinates) of the `\Vector`, `\Line`, and `\Curve` commands is appreciated. The first `\oval` command draws a normal ellipse with a thick line (using the `\thicklines` command), while the second `\oval` command draws a rectangle with rounded corners and thin-line borders (using the `\thinlines` command). Finally, we set the line width to 3pt with the `\linethickness` command and show the effect on circles and lines.




```
\usepackage{pspicture}\usepackage{graphpap}
\begin{picture}(150,120)
\graphpaper(0,0)(150,120)
\arrowlength{4pt} \put(150,00){\vector(-8,1){60}}
\arrowlength{8pt} \put(150,50){\Vector(-30,50)}
\put(60,20){\Line(90,20)}
\put(60,20){\Curve(90,20){2}}
\put(60,20){\Curve(90,20){-2}}
\thicklines \put(50,80){\oval(100,70)}
\thinlines  \put(50,80){\oval[10](100,70)}
\linethickness{3pt}
            \put(10,20){\circle{20}}
            \put(10,20){\line(10,1){30}}
\end{picture}
```

10-4-4

10.4.4 The Portable Document Format

Adobe's Portable Document Format (PDF) [5] is a direct descendant of the PostScript language. Whereas PostScript is a full-blown programming language, PDF is a second-generation, more light-weight graphics language optimized for faster download and display. Most of the advantages of PostScript remain: PDF guarantees page fidelity, down to the smallest glyph or piece of white space, while being portable across different computer platforms. For these reasons, PDF is being used ever more frequently in the professional printing world as a replacement for PostScript. Moreover, all present-day browsers will embed or display PDF material, alongside HTML, using plug-in technology.

The main differences between PostScript and PDF are the following:

- There are no built-in programming language functions: for example, PDF in general cannot calculate values.
- PDF guarantees full page independence by clearly separating resources from page objects.
- PDF files are compact and fully searchable.
- Interactive hyperlinks make PDF files easy to navigate.
- PDF's security features allow PDF documents to have special access rights and digital signatures applied.
- Font outlines need not be included in the file, because PDF files carry sufficient font information information to allow PDF-enabled applications (e.g., Adobe's Acrobat Reader) to mimic the appearance of a font.
- PDF has advanced compression features to keep the size of PDF files small. Moreover, `.png`, `.jpeg`, and `.gif` images can be inserted directly.
- PDF 1.4 and later versions support a transparent imaging model (PostScript uses an opaque model) and feature multimedia support.
- PDF 1.4 and later versions introduce tagged PDF, a stylized form of PDF that contains information on content and structure. Tagged PDF lets applications extract and reuse page data (text, graphics, images). For instance, tagged PDF allows text to reflow for display on handheld devices, such as Palm OS or Pocket PC systems.
- PDF 1.5, released at the end of 2003, includes features for further optimizing multimedia delivery.

PDF can be viewed and printed on many different computer platforms by downloading and installing Adobe's Acrobat Reader.[1] Other PDF viewers exist as well. The best-known free ones are ghostscript,[2] which can also produce PDF from PostScript, and Xpdf.[3]

[1] Freely downloadable from `http://www.adobe.com/products/acrobat/readermain.html`.
[2] See `http://www.cs.wisc.edu/~ghost/`.
[3] See `http://www.foolabs.com/xpdf/home.html`.

Generating PDF directly from TeX

If you have a PostScript file generated from a LaTeX source, you can convert it to PDF by using a "distiller" program. Adobe's Acrobat Distiller is the best known and most sophisticated of these programs, but ghostscript (and ImageMagick's convert, which is built on it) also performs well.

To generate PDF directly without going through the `dvi`-generating step, we have pdfTeX (see below) and MicroPress's VTeX,[1] which has its own direct PDF-generating TeX engine. If you already have a `dvi` file, you can use Mark Wicks's dvipdfm `dvi` driver.[2]

Hán Thế Thánh's pdfTeX is an extension of TeX that creates PDF directly from TeX source files [161]. It also enhances the typesetting capabilities of TeX in some interesting areas [158,159]. Since 2002 pdfTeX has been part of the standard TeX distributions.

The pdfTeX program lets you include annotations, hyperlinks, and bookmarks in the generated PDF output file. It can work with TrueType fonts and supports the inclusion of pictures in `.png` and `.jpeg` formats. The most common technique, the inclusion of Encapsulated PostScript figures, has been replaced by PDF inclusion in this program. EPS files can be converted to PDF by ImageMagick's convert utility, eps2pdf (both of which call ghostscript internally), Acrobat Distiller, or other PostScript-to-PDF converters.

Navigation is an important aspect of PDF documents. The hyperref package [56, Chapter 2] developed by Sebastian Rahtz and Heiko Oberdiek extends the functionality of the LaTeX cross-referencing commands (including the table of contents, bibliographies, and so on) to produce `\special` commands that a `dvi` driver or pdfTeX can turn into hypertext links. The hyperref package also provides new commands to allow the user to write ad hoc hypertext links, including those to external documents and URLs.

Because PDF lacks programming language commands, it cannot deal with general raw PostScript commands, such as those used by the pstricks package [57, Chapter 4]. Thus, these commands are not supported.[3]

The standard LaTeX graphics and color packages have a `pdftex` option, which allow you to use normal color, text rotation, and graphics inclusion commands. The implementation of graphics inclusion makes sure that however often a graphic is used (even if it is used at different scales or transformed in different ways), it is embedded only once.

Producing correct PostScript or PDF

Getting correct PostScript or PDF output from LaTeX systems can sometimes be quite difficult. Michael Shell, in the context of the IEEEtran document class files, but independent of them, has developed the "testflow" diagnostic suite. A test file

[1] See `http://www.micropress-inc.com/`.

[2] See `http://gaspra.kettering.edu/dvipdfm/`.

[3] General PostScript commands *can* be used with MicroPress's VTeX, which has a built-in PostScript interpreter.

`testflow.tex` is first compiled on the user's system. Next, a PostScript version, `testflow.ps`, and a PDF version, `testflow.pdf`, for the output are produced and printed on the output device for comparison to reference files. The input test file is designed to test the various components of LaTeX's "print work flow". Its purpose is to provide helpful information to assist users in getting their LaTeX system configured correctly so as to produce good PostScript and PDF output.[1]

10.4.5 Scalable Vector Graphics

Since the mid-1990s, the World Wide Web and the general availability of the personal computer have made the generation, maintenance, and dissemination of electronic documents worldwide cheap, easy, and fast. Moreover, the development of the XML family of standards and the ubiquity of platform-independent scripting languages allow one to save and handle huge amounts of electronically stored information and to transform it into various customizable presentation forms.

For LaTeX documents, a variety of techniques are available to transform them into PDF, XHTML, or XML so that the information can be made available on the web. Thus, LaTeX can continue to play a major role in the integrated worldwide cyberspace, in particular for scientific documents, and especially in areas where fine typesetting is a must.

After a short introduction to Scalable Vector Graphics (SVG), we explain succinctly how LaTeX-encoded information can be encoded into an SVG-format (see [58] for more detail).

SVG for portable graphics on the web

As the web has grown in popularity and complexity, users and content providers have sought ever better, more precise, and more scalable graphical rendering—not just the low-resolution `.gif` or `.png` images that are commonly used in today's web pages. To address this need, the World Wide Web Consortium published the SVG Recommendation, whose current version is 1.1.[2]

SVG is an open-standard vector graphics language for describing two-dimensional graphics using XML syntax. It lets you produce web pages containing high-resolution computer graphics.

As an XML instance, SVG consists of Unicode text. It features the usual vector graphics functions. Its fundamental primitive is the *graphics object*, whose model contains the following:

- Graphics paths consisting of polylines, Bézier curves, and other elements:
  - Simple or compound, closed or open
  - (Gradient) filled, (gradient) stroked

[1]Detailed instructions and a detailed explanation available at CTAN: `macros/latex/contrib/IEEEtran/testflow/testflow_doc.txt`.

[2]*Scalable Vector Graphics (SVG) 1.1 Specification*, available at `http://www.w3.org/TR/SVG11/`, was published on January 14, 2003.

- Can be used for clipping
- Can be used for building common geometric shapes

- Patterns and markers
- Templates and symbol libraries
- Transformations:
  - Default coordinate system: x is right, y is down,[1] the unit is one pixel
  - Viewport maps an area in world coordinates to an area on screen
  - Transformations alter the coordinate system (2×3 transformation matrix for computers; translate, rotate, scale, skew for humans)
  - Can be nested
- Inclusion of bitmap or raster images
- Clipping, filter, and raster effects; alpha masks
- Animations, scripts, and extensions
- Groupings and styles
- SVG fonts (independent from fonts installed on the system)

The W3C SVG web site (`http://www.w3.org/Graphics/SVG`) is a good first source of information and has a lot of pointers to other sites.

Transforming a LaTeX document into an SVG document

If one has a pure LaTeX source document (i.e., one that includes no EPS files, nor uses any extensions that need TeX `\special` commands), the `dvi` file can be translated into SVG with Adrian Frischauf's dvi2svg.[2]

We interacted with the `dvi2svg` Java library via a small UN*X script called `dvi2svg.sh`, whose use is as follows:

```
> dvi2svg.sh
Usage: dvi2svg.sh [options] [DVIFILE]
Options:
 -o [FILENAME] : Specify an output filename prefix. If not
                 set, dvi2svg will take the input filename.
 -d : set the debug mode to on(1)/off(0 default)
```

An example of the use of the `dvi2svg` program is the translation of two examples in this chapter into SVG. We compile the LaTeX file `svgexa.tex` and then run

[1]The reference point of the display area is the upper-left corner. For PostScript, where y runs upward, the reference point of the page is the lower-left corner.

[2]See `http://www.activemath.org/~adrianf/dvi2svg/`. The dvi2svg program includes SVG font outlines for the characters referenced in the `dvi` file. SVG font instances were generated for all standard Computer Modern and LaTeX fonts and come with the dvi2svg distribution.

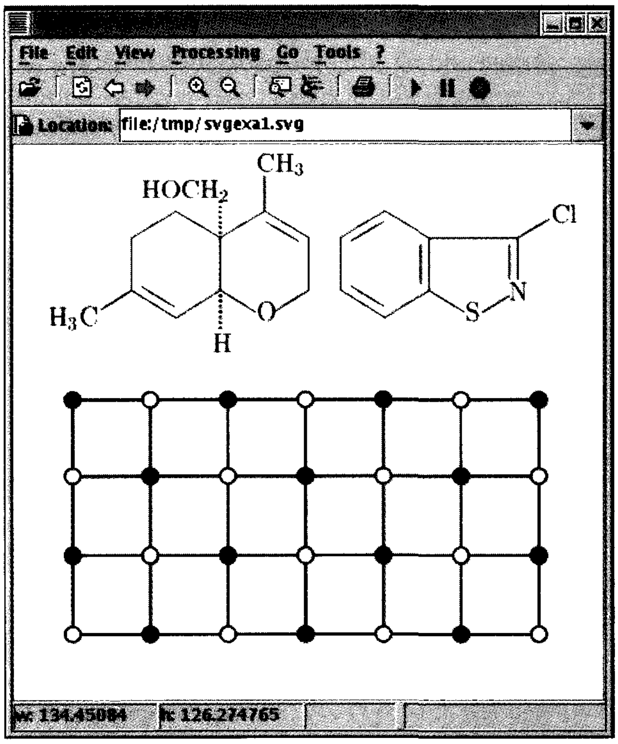


Figure 10.3: SVG generated from a dvi file

dvi2svg.sh on the generated dvi file to obtain the SVG file svgexa1. (If the dvi file contains more than one page several output files are generated.)

```
> dvi2svg.sh svgexa.dvi -o svgexa
DEBUG from converter.DviToSvg => Converting file: svgexa.dvi
DEBUG from converter.DviToSvg => Writing result to: svgexa
DEBUG from converter.DviToSvg => Reader has been created
DEBUG from converter.DviToSvg => Writer has been created
Converting ...............FINISHED
> ls -l svgexa*.svg
-rw-rw-r--    1 goossens     23792 Jun 25 19:44 svgexa1.svg
```

Figure 10.3 shows the generated SVG file as viewed with the squiggle program.[1] For more complex LaTeX files (in particular, those with EPS or PDF inclusions) you can first generate a PostScript file with dvips, and then use Wolfgang Glunz's pstoedit program (see [58] for an explanation of how it works).

[1]The squiggle SVG browser is part of the Apache Batik distribution (http://xml.apache.org/batik). SVG can also be viewed with Adobe's browser plugin svgview (http://www.adobe.com/svg).

CHAPTER 11

Index Generation

To find a topic of interest in a large document, book, or reference work, you usually turn to the table of contents or, more often, to the index. Therefore, an index is a very important part of a document, and most users' entry point to a source of information is precisely through a pointer in the index. You should, therefore, plan an index and develop it along with the main text [38]. For reasons of consistency, it is beneficial, with the technique discussed below, to use special commands in the text to always print a given keyword in the same way in the text and the index throughout the whole document.

This chapter first reviews the basic indexing commands provided by standard LaTeX, and explains which tools are available to help you build a well-thought-out index. The *LaTeX Manual* itself does not contain a lot of information about the syntax of the `\index` entries. However, several articles in *TUGboat* deal with the question of generating an index with TeX or LaTeX [47, 162, 163]. The syntax described in Section 11.1 is the one recognized by *MakeIndex* [37, 103] and `xindy` [71, 76, 152], the most widely used index preparation programs.

Section 11.2 describes how the *MakeIndex* processor is used. The interpretation of the input file and the format of the output file are controlled by style parameters. Section 11.2.4 lists these parameters and gives several simple examples to show how changing them influences the typeset result.

Section 11.3 presents `xindy`, an alternative to *MakeIndex*. It's preferable to use this program whenever you have non-English documents or other special demands, such as production of technical indexes. The `xindy` program provides total flexibility for merging and sorting index entries, and for arbitrary formatting of references.

The final section describes several LaTeX packages to enhance the index and to create multiple indexes, which will be discussed with the help of an example.

① A raw index (.idx file) is generated by running LaTeX.

② The raw index, together with some optional style information (.ist file), is used as input to the index processor, which creates an alphabetized index (.ind file) and a transcript (.ilg file).

③ The index (.ind file) is read by LaTeX to give the final typeset result.

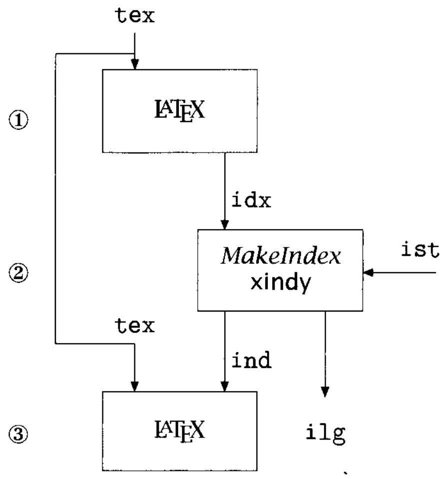


Figure 11.1: The sequential flow of index processing and the various auxiliary files used by LaTeX and external index processors

The process of generating an index is shown schematically in Figure 11.1. The steps for generating an index with LaTeX and either *MakeIndex* or xindy are illustrated in this figure.

Figure 11.2 on the next page shows, with an example, the various steps involved in transforming an input file into a typeset index. It also shows, in somewhat more detail, which files are involved in the index-generating process. Figure 11.2(a) shows some occurrences of index commands (\index) in the document source, with corresponding pages listed on the left. Figure 11.2(b) shows a raw index .idx file generated by LaTeX. File extensions may differ when using multiple indexes or glossaries. After running the .idx file through the index processor, it becomes an alphabetized index .ind file with LaTeX commands specifying a particular output format [Figure 11.2(c)]. The typeset result after formatting with LaTeX is shown in Figure 11.2(d).

LaTeX and *MakeIndex*, when employed together, use several markup conventions to help you control the precise format of the output. The xindy program has a *MakeIndex* compatibility mode that supports the same format. In Section 11.1, which describes the format of the \index command, we always use the default settings.

11.1 Syntax of the index entries

This section describes the default syntax used to generate index entries with LaTeX and either *MakeIndex* or xindy. Different levels of complexity are introduced progressively, showing, for each case, the input file and the generated typeset output.

Page vi:	`\index{animal}`	`\indexentry{animal}{vi}`
Page 5:	`\index{animal}`	`\indexentry{animal}{5}`
Page 6:	`\index{animal}`	`\indexentry{animal}{6}`
Page 7:	`\index{animal}`	`\indexentry{animal}{7}`
Page 11:	`\index{animalism|see{animal}}`	`\indexentry{animalism|see{animal}}{11}`
Page 17:	`\index{animal@\emph{animal}}`	`\indexentry{animal@\emph{animal}}{17}`
	`\index{mammal|textbf}`	`\indexentry{mammal|textbf}{17}`
Page 26:	`\index{animal!mammal!cat}`	`\indexentry{animal!mammal!cat}{26}`
Page 32:	`\index{animal!insect}`	`\indexentry{animal!insect}{32}`

(a) The input file *(b) The `.idx` file*

```
\begin{theindex}
  \item animal, vi, 5-7
    \subitem insect, 32
    \subitem mammal
      \subsubitem cat, 26
  \item \emph{animal}, 17
  \item animalism, \see{animal}{11}
  \indexspace
  \item mammal, \textbf{17}
\end{theindex}
```

(c) The `.ind` file

animal, vi, 5–7
  insect, 32
  mammal
    cat, 26
animal, 17
animalism, *see* animal

mammal, **17**

(d) The typeset output

Figure 11.2: Stepwise development of index processing

Figures 11.3 and 11.4 on page 656 show the input and generated output of a small LaTeX document, where various simple possibilities of the `\index` command are shown, together with the result of including the showidx package (see Section 11.4.2). To make the index entries consistent in these figures (see Section 11.1.7), the commands `\Com` and `\Prog` were defined and used. The index-generating environment `theindex` has been redefined to get the output on one page (Section 11.4.1 explains how this can be done).

Generating the raw index

After introducing the necessary `\index` commands in the document, we want to generate the index to be included once again in the LaTeX document on a subsequent run. If the main file of a document is `main.tex`, for example, then the following changes should be made to that file:

- Include the makeidx package with a `\usepackage` command.
- Put a `\makeindex` command in the document preamble.
- Put a `\printindex` command where the index is to appear—usually at the end, right before the `\end{document}` command.

You then run LaTeX on the entire document, causing it to generate the file `main.idx`, which we shall call the `.idx` file.

11.1.1 Simple index entries

Each `\index` command causes LaTeX to write an entry in the `.idx` file. The following example shows some simple `\index` commands, together with the index entries that they produce. The page number refers to the page containing the text where the `\index` command appears. As shown in the example below, duplicate commands on the same page (such as `\index{stylistic}` on page 23) produce only one "23" in the index.

style, 14	Page iii:	`\index{style}`
style , 16	Page xi:	`\index{Stylist}`
style, iii, 12	Page 12:	`\index{style}`
style , 15		`\index{styles}`
style file, 34	Page 14:	`\index{ style}`
styles, 12	Page 15:	`\index{style }`
Stylist, xi	Page 16:	`\index{ style }`
stylist, 34	Page 23:	`\index{stylistic}`
stylistic, 23		`\index{stylistic}`
	Page 34:	`\index{style file}`
		`\index{stylist}`

Spaces can be harmful

Pay particular attention to the way spaces are handled in this example. Spaces inside `\index` commands are written literally to the output `.idx` file and, by default, are treated as ordinary characters by *MakeIndex*, which places them in front of all letters. In the example above, look at the `style` entries on pages 14 and 16. The leading spaces are placed at the beginning of the index and on two different lines because the trailing blank on page 16 lengthens the string by one character. We end up with four different entries for the same term, an effect that was probably not desired. It is therefore important to eliminate such spurious spaces from the `\index` commands when you use *MakeIndex*. Alternatively, you can specify the `-c` option when running the index processor. This option suppresses the effect of leading and trailing blanks (see Sections 11.2.2 and 11.3.1). Another frequently encountered error occurs when the same English word is spelled inconsistently with initial lowercase and uppercase letters (as with `Stylist` on page `xi`), leading to two different index entries. Of course, this behavior is wanted in languages like German, where "Arm" (arm) and "arm" (poor) are really two completely different words. In English, such spurious double entries should normally be eliminated.

If you use xindy, space compression is done automatically. Furthermore, xindy supports international indexing and thus correctly and automatically handles case sensitivity in a language-specific way. Therefore, with xindy you won't encounter the problems mentioned above.

11.1.2 Generating subentries

A maximum of three levels of index entries (main, sub, and subsub entries) are available. To produce such entries, the argument of the `\index` command should contain both the main entries and subentries, separated by a ! character. This character can be redefined in the *MakeIndex* style file (see Table 11.1 on page 660).

box, 21	Page 3:	`\index{dimensions!rule!width}`
dimensions of, 33	Page 5:	`\index{box!parameters}`
parameters, 5	Page 9:	`\index{dimensions!table}`
dimensions	Page 12:	`\index{dimensions!rule!height}`
figure, 12		`\index{dimensions!figure}`
rule	Page 21:	`\index{box}`
height, 12	Page 33:	`\index{box!dimensions of}`
width, 3		
table, 9		

11.1.3 Page ranges and cross-references

You can specify a page range by putting the command `\index{...|(}` at the beginning of the range and the command `\index{...|)}` at the end of the range. Page ranges should span a homogeneous numbering scheme (e.g., Roman and Arabic page numbers cannot fall within the same range). Note that *MakeIndex* and xindy do the right thing when both ends of a page range fall on the same page, or when an entry falls inside an active range.

You can also generate cross-reference index entries without page numbers by using the `see` encapsulator. Because the "see" entry does not print any page number, the commands `\index{...|see{...}}` can be placed anywhere in the input file *after* the `\begin{document}` command. For practical reasons, it is convenient to group all such cross-referencing commands in one place.

fonts	Page ii:	`\index{table|(}`
Computer Modern, 13–25	Page xi:	`\index{table|)}`
math, *see* math, fonts	Page 5:	`\index{fonts!PostScript|(}`
PostScript, 5		`\index{fonts!PostScript|)}`
table, ii–xi, 14	Page 13:	`\index{fonts!Computer Modern|(}`
	Page 14:	`\index{table}`
	Page 17:	`\index{fonts!math|see{math, fonts}}`
	Page 21:	`\index{fonts!Computer Modern}`
	Page 25:	`\index{fonts!Computer Modern|)}`

11.1.4 Controlling the presentation form

Sometimes you may want to sort an entry according to a key, while using a different visual representation for the typesetting, such as Greek letters, mathematical symbols, or specific typographic forms. This function is available with the syntax *key@visual*, where *key* determines the alphabetical position and the string *visual* produces the typeset text of the entry.

delta, 14
δ, 23
delta wing, 16
flower, 19
ninety, 26
xc, 28
ninety-five, 5
`tabular` environment, 23

```
Page 5:   \index{ninety-five}
Page 14:  \index{delta}
Page 16:  \index{delta wing}
Page 19:  \index{flower@\textbf{flower}}
Page 23:  \index{delta@$\delta$}
          \index{tabular@\texttt{tabular} environment}
Page 26:  \index{ninety}
Page 28:  \index{ninety@xc}
```

For some indexes, certain page numbers should be formatted specially. For example, an italic page number might indicate a primary reference, or an *n* after a page number might denote that the item appears in a footnote on that page. *MakeIndex* allows you to format an individual page number in any way you want by using the encapsulator syntax specified by the `|` character. What follows the `|` sign will "encapsulate" or enclose the page number associated with the index entry. For instance, the command `\index{keyword|xxx}` will produce a page number of the form `\xxx{`*n*`}`, where *n* is the page number in question. Similarly, the commands `\index{keyword|(xxx}` and `\index{keyword|)xxx}` will generate a page range of the form `\xxx{`*n-m*`}`.

Preexisting commands (like `\textit` in the example below) or user commands can be used to encapsulate the page numbers. As an example, a document containing the command definition

```
\newcommand\nn[1]{#1n}
```

would yield something like this:

tabular, **ii**, *21*, 22n
tabbing, 7, *34–37*

```
Page ii:  \index{tabular|textbf}
Page 7:   \index{tabbing}
Page 21:  \index{tabular|textit}
Page 22:  \index{tabular|nn}
Page 34:  \index{tabbing|(textit}
Page 37:  \index{tabbing|)textit}
```

The `see` encapsulator is a special case of this facility, where the `\see` command is predefined by the `makeidx` package.

11.1.5 Printing special characters

To typeset one of the characters having a special meaning to *MakeIndex* or `xindy` (`!`, `"`, `@`, or `|`)[1] in the index, precede it with a `"` character. More precisely, any character is said to be quoted if it follows an unquoted `"` that is not part of a `\"` command. The latter case allows for umlaut characters. Quoted `!`, `@`, `"`, and `|` characters are treated like ordinary characters, losing their special meaning. The `"` preceding a quoted character is deleted before the entries are alphabetized.

[1]As noted earlier, in *MakeIndex* other characters can be substituted for the default ones and carry a special meaning. This behavior is explained on page 662.

@ sign, 2
|, *see* vertical bar
exclamation (!), 4
  Ah!, 5
Mädchen, 3
quote ("), 1
" sign, 1

```
        \index{bar@\texttt{"|}|see{vertical bar}}
Page 1: \index{quote (\verb+""+)}
        \index{quote@\texttt{""} sign}
Page 2: \index{atsign@\texttt{"@} sign}
Page 3: \index{maedchen@M\"{a}dchen}
Page 4: \index{exclamation ("!)}
Page 5: \index{exclamation ("!)!Ah"!}
```

11.1.6 Creating a glossary

LaTeX also has a `\glossary` command for making a glossary. The `\makeglossary` command produces a file with an extension of `.glo`, which is similar to the `.idx` file for the `\index` commands. LaTeX transforms the `\glossary` commands into `\glossaryentry` entries, just as it translates any `\index` commands into `\indexentry` entries.

MakeIndex can also handle these glossary commands, but you must change the value for some of the style file keywords, as shown in the style file `myglossary.ist`.

```
% MakeIndex style file myglossary.ist
keyword    "\\glossaryentry"            % keyword for glossary entry
preamble   "\n \\begin{theglossary}\n"  % Begin glossary entries
postamble  "\n\n \\end{theglossary}\n"  % End   glossary entries
```

In addition, you have to define a suitable `theglossary` environment.

11.1.7 Defining your own index commands

As was pointed out in the introduction, it is very important to use the same visual representation for identical names or commands throughout a complete document, including the index. You therefore can define user commands, which always introduce similar constructs in the same way into the text and the index.

For example, you can define the command `\Index`, whose argument is entered at the same time in the text and in the index.

```
\newcommand\Index[1]{#1\index{#1}}
```

As explained in more detail below, you must be careful that the argument of such a command does not contain expandable material (typically control sequences) or spurious blanks. In general, for simple terms like single words, there is no problem and this technique can be used. You can even go one step further and give a certain visual representation to the entry—for instance, typesetting it in a typewriter font.

```
\newcommand\Indextt[1]{\texttt{#1}\index{#1@\texttt{#1}}}
```

Finally, you can group certain terms by defining commands that have a generic meaning. For instance, LaTeX commands and program names could be treated with special commands, as in the following examples:

```
\newcommand\bs{\symbol{'134}} % print backslash in typewriter OT1/T1
\newcommand\Com[1]{\texttt{\bs#1}\index{#1@\texttt{\bs#1}}}
\newcommand\Prog[1]{\texttt{#1}\index{#1@\texttt{#1} program}}
```

The `\Com` command adds a backslash to the command's name in both text and index, simplifying the work of the typist. The `\bs` command definition is necessary, because `\textbackslash` would be substituted in an `OT1` font encoding context, as explained in Section 7.3.5 on page 346. At the same time, commands will be ordered in the index by their names, with the \-character being ignored during sorting. Similarly, the `\Prog` command does not include the `\texttt` command in the alphabetization process, because entries like `\index{\texttt{`*key*`}}` and `\index{`*key*`}` would then result in different entries in the index.

11.1.8 Special considerations

When an `\index` command is used directly in the text, its argument is expanded only when the index is typeset, not when the `.idx` file is written. However, when the `\index` command is contained in the argument of another command, characters with a special meaning to TeX, such as \, must be properly protected against expansion. This problem is likely to arise when indexing items in a footnote, or when using commands that put their argument in the text and enter it at the same time in the index (see the discussion in Section 11.1.7). Even in this case, robust commands can be placed in the "`@`" part of an entry, as in `\index{rose@\textit{rose}}`, but fragile commands must be protected with the `\protect` command.

As with every argument of a command you need to have a matching number of braces. However, because `\index` allows special characters like `%` or `\` in its argument if the command is used in main text, the brace matching has an anomaly: braces in the commands `\{` and `\}` take part in the matching. Thus, you cannot write `\index{\{}` or something similar.

11.2 makeindex—A program to format and sort indexes

In the previous section we showed examples where we ran the *MakeIndex* program using its default settings. In this section we will first take a closer look at the *MakeIndex* program, and then discuss ways of changing its behavior.

11.2.1 Generating the formatted index

To generate the formatted index, you should run the *MakeIndex* program by typing the following command (where `main` is the name of the input file):

```
makeindex main.idx
```

This produces the file `main.ind`, which will be called the `.ind` file here. If *MakeIndex* generated no error messages, you can now rerun LaTeX on the document and the index will appear. (You can remove the `\makeindex` command if you do not want to regenerate the index.) Page 658 describes what happens at this point if there are error messages.

In reading the index, you may discover additional mistakes. These should be corrected by changing the appropriate `\index` commands in the document and regenerating the `.ind` file (rerunning LaTeX *before* and *after* the last step).

An example of running *MakeIndex* is shown below. The `.idx` file, `main.idx`, is generated by a first LaTeX run on the input shown in Figure 11.3 on the next page. You can clearly see that two files are written—namely, the ordered `.ind` index file for use with LaTeX, called `main.ind`, and the index `.ilg` log file, called `main.ilg`, which (in this case) will contain the same text as the output on the terminal. If errors are encountered, then the latter file will contain the line number and error message for each error in the input stream. Figure 11.4 on the following page shows the result of the subsequent LaTeX run. The example uses the showidx package for controlling the index (see Section 11.4.2).

```
> makeindex main
This is makeindex, version 2.13 [07-Mar-1997] (using kpathsea).
Scanning input file main.idx....done (8 entries accepted, 0 rejected).
Sorting entries....done (24 comparisons).
Generating output file main.ind....done (19 lines written, 0 warnings).
Output written in main.ind.
Transcript written in main.ilg.
```

11.2.2 Detailed options of the *MakeIndex* program

The syntax of the options of the *MakeIndex* program are described below:

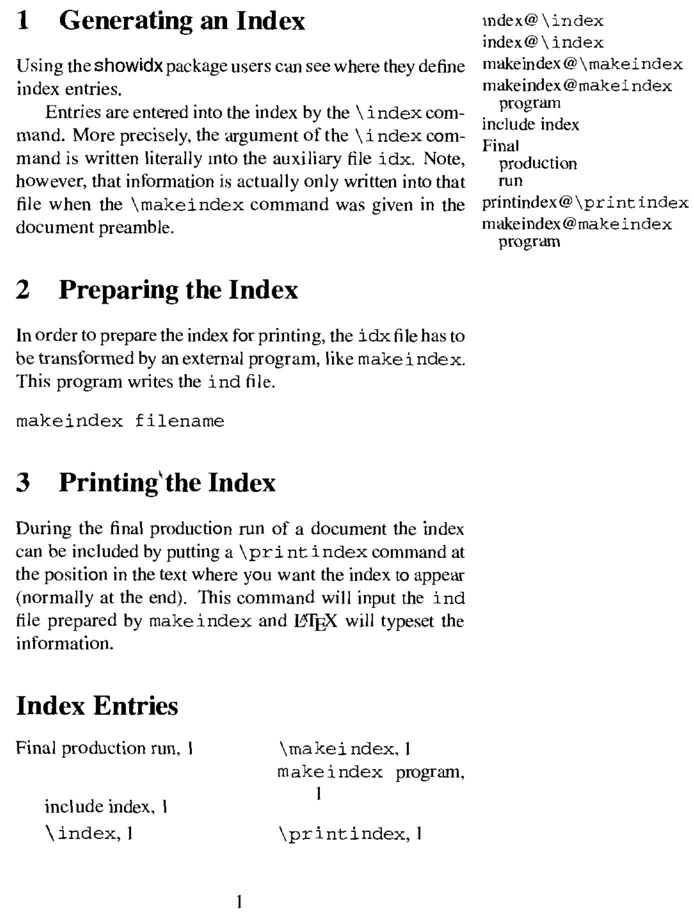
makeindex [-ciglqr] [-o *ind*] [-p *no*] [-s *sty*] [-t *log*] [*idx0 idx1* ...]

`-c` Enable blank compression. By default, every blank counts in the index key. The `-c` option ignores leading and trailing blanks and tabs and compresses intermediate ones to a single space.

`-i` Use standard input (`stdin`) as the input file. When this option is specified and `-o` is not, output is written to standard output (`stdout`, the default output stream).

```
\documentclass{article}
\usepackage{makeidx,showidx}
\newcommand\bs{\symbol{'134}}% print backslash
\newcommand\Com[1]{\texttt{\bs#1}%
                  \index{#1@\texttt{\bs#1}}}
\newcommand\Prog[1]{\texttt{#1}%
                  \index{#1@\texttt{#1} program}}
\makeindex
\begin{document}
\section{Generating an Index}
Using the \textsf{showidx} package users can
see where they define index entries.
\par Entries are entered into the index by the
\Com{index} command. More precisely, the argument
of the \Com{index} command is written literally into
the auxiliary file \texttt{idx}. Note, however, that
information is only actually written into that file
when the \Com{makeindex} command was given in the
document preamble.

\section{Preparing the Index}
In order to prepare the index for printing, the
\texttt{idx} file has to be transformed by an external
program, like \Prog{makeindex}.
This program writes the \texttt{ind} file.
\begin{verbatim}
makeindex filename
\end{verbatim}

\section{Printing the Index}\index{Final production run}
During the final production run of a document the
index can be \index{include index}included by putting
a \Com{printindex} command at the position in the text
where you want the index to appear (normally at the
end).This command will input the \texttt{ind} file
prepared by the \Prog{makeindex} and \LaTeX{} will
typeset the information.
\printindex
\end{document}
```

Figure 11.3: Example of `\index` commands and the showidx package. This file is run through LaTeX once, then the index processor is executed and LaTeX is run a second time.



1 Generating an Index

Using the showidx package users can see where they define index entries.

Entries are entered into the index by the `\index` command. More precisely, the argument of the `\index` command is written literally into the auxiliary file `idx`. Note, however, that information is actually only written into that file when the `\makeindex` command was given in the document preamble.

index@`\index`
index@`\index`
makeindex@`\makeindex`
makeindex@`makeindex` program
include index
Final production run
printindex@`\printindex`
makeindex@`makeindex` program

2 Preparing the Index

In order to prepare the index for printing, the `idx` file has to be transformed by an external program, like `makeindex`. This program writes the `ind` file.

`makeindex filename`

3 Printing the Index

During the final production run of a document the index can be included by putting a `\printindex` command at the position in the text where you want the index to appear (normally at the end). This command will input the `ind` file prepared by `makeindex` and LaTeX will typeset the information.

Index Entries

Final production run, 1

include index, 1
`\index`, 1

`\makeindex`, 1
`makeindex` program, 1

`\printindex`, 1

1

Figure 11.4: This figure shows the index generated by the example input of Figure 11.3. All index entries are shown in the margin, so it is easy to check for errors or duplications.

`-g` Employ German word ordering in the index, following the rules given in German standard DIN 5007. In this case the normal precedence rule of *MakeIndex* for word ordering (symbols, numbers, uppercase letters, lowercase letters) is replaced by the German word ordering (symbols, lowercase letters, uppercase letters, numbers). Additionally, this option enables *MakeIndex* to recognize the German TEX commands `"a`, `"o`, `"u`, and `"s` as `ae`, `oe`, `ue`, and `ss`, respectively, for sorting purposes. The quote character must be redefined in a style file (see page 662); otherwise, you will get an error message and *MakeIndex* will abort. Note that not all versions of *MakeIndex* recognize this option.

`-l` Use letter ordering. The default is word ordering. In word ordering, a space comes before any letter in the alphabet. In letter ordering, spaces are ignored. For example, the index terms "point in space" and "pointing" will be alphabetized differently in letter and word ordering.

`-q` Operate in quiet mode. No messages are sent to the error output stream (`stderr`). By default, progress and error messages are sent to `stderr` as well as the transcript file. The `-q` option disables the `stderr` messages.

`-r` Disable implicit page range formation. By default, three or more successive pages are automatically abbreviated as a range (e.g., `1-5`). The `-r` option disables this default, making explicit range operators the only way to create page ranges.

`-o` *ind* Take *ind* as the output index file. By default, the file name base of the first input file *idx0* concatenated with the extension `.ind` is used as the output file name.

`-p` *no* Set the starting page number of the output index file to *no*. This option is useful when the index file is to be formatted separately. Other than pure numbers, three special cases are allowed for *no*: any, odd, and even. In these special cases, the starting page number is determined by retrieving the last page number from the `.log` file of the last LATEX run. The `.log` file name is determined by concatenating the file name base of the first raw index file (*idx0*) with the extension `.log`. The last source page is obtained by searching backward in the log file for the first instance of a number included in square brackets. If a page number is missing or if the `.log` file is not found, no attempt will be made to set the starting page number. The meaning of each of the three special cases follows:

`any` The starting page is the last source page number plus one.

`odd` The starting page is the first odd page following the last source page number.

`even` The starting page is the first even page following the last source page number.

-s *sty* Take *sty* as the style file. There is no default for the style file name. The environment variable INDEXSTYLE defines where the style file resides.

-t *log* Take *log* as the transcript file. By default, the file name base of the first input file *idx0* concatenated with the extension .ilg is used as the transcript file name.

11.2.3 Error messages

MakeIndex displays on the terminal how many lines were read and written and how many errors were found. Messages that identify errors are written in the transcript file, which, by default, has the extension .ilg. *MakeIndex* can produce error messages when it is reading the .idx file or when it is writing the .ind file. Each error message identifies the nature of the error and the number of the line where the error occurred in the file.

Errors in the reading phase

In the reading phase, the line numbers in the error messages refer to the positions in the .idx file being read.

`Extra '!' at position ...`
The \index command's argument has more than two unquoted ! characters. Perhaps some of them should be quoted.

`Extra '@' at position ...`
The \index command argument has two or more unquoted @ characters with no intervening !. Perhaps one of the @ characters should be quoted.

`Extra '|' at position ...`
The \index command's argument has more than one unquoted | character. Perhaps the extras should be quoted.

`Illegal null field`
The \index command argument does not make sense because some string is null that shouldn't be. The command \index{!funny} will produce this error, since it specifies a subentry "funny" with no entry. Similarly, the command \index{@funny} is incorrect, because it specifies a null string for sorting.

`Argument ... too long (max 1024)`
The document contained an \index command with a very long argument. You probably forgot the right brace that should delimit the argument.

Errors in the writing phase

In the writing phase, line numbers in the error messages refer to the positions in the .ind file being written.

`Unmatched range opening operator`
An \index{...|(} command has no matching \index{...|)} command following it. The "..." in the two commands must be completely identical.

`Unmatched range closing operator`
An \index{...|)} command has no matching \index{...|(} command preceding it.

`Extra range opening operator`
: Two `\index{...|(}` commands appear in the document with no intervening command `\index{...|)}`.

`Inconsistent page encapsulator ... within range`
: *MakeIndex* has been instructed to include a page range for an entry and a single page number within that range is formatted differently—for example, by having an `\index{cat|see{animals}}` command between an `\index{cat|(}` command and an `\index{cat|)}` command.

`Conflicting entries`
: *MakeIndex* thinks it has been instructed to print the same page number twice in two different ways. For example, the command sequences `\index{lion|see{...}}` and `\index{lion}` appear on the same page.

MakeIndex can produce a variety of other error messages indicating that something is seriously wrong with the `.idx` file. If you get such an error, it probably means that the `.idx` file was corrupted in some way. If LaTeX did not generate any errors when it created the `.idx` file, then it is highly unlikely to have produced a bad `.idx` file. If, nevertheles, this does happen, you should examine the `.idx` file to establish what went wrong.

11.2.4 Customizing the index with *MakeIndex*

MakeIndex ensures that the formats of the input and output files do not have to be fixed, but they can be adapted to the needs of a specific application. To achieve this format independence, the *MakeIndex* program is driven by a style file, usually characterized with a file extension of `.ist` (see also Figure 11.1 on page 648). This file consists of a series of keyword/value pairs. These keywords can be divided into input and output style parameters. Table 11.1 on the following page describes the various keywords and their default values for the programming of the input file. This table shows, for instance, how to modify the index level separator (`level`, with ! as default character value). Table 11.2 on page 661 describes the various keywords and their default values for steering the translation of the input information into LaTeX commands. This table explains how to define the way the various levels are formatted (using the `item` series of keywords). Examples will show in more detail how these input and output keywords can be used in practice. *MakeIndex* style files use UN*X string syntax, so you must enter \\ to get a single \ in the output.

In the following sections we show how, by making just a few changes to the values of the default settings of the parameters controlling the index, you can customize the index.

A stand-alone index

The example style `mybook.ist` (shown below) defines a stand-alone index for a book, where "stand-alone" means that it can be formatted independently of the

Keyword	*Default Value*	*Description*
`keyword` (*s*)	`"\\indexentry"`	Command telling *MakeIndex* that its argument is an index entry
`arg_open` (*c*)	`'{'`	Argument opening delimiter
`arg_close` (*c*)	`'}'`	Argument closing delimiter
`range_open` (*c*)	`'('`	Opening delimiter indicating the beginning of an explicit page range
`range_close` (*c*)	`')'`	Closing delimiter indicating the end of an explicit page range
`level` (*c*)	`'!'`	Delimiter denoting a new level of subitem
`actual` (*c*)	`'@'`	Symbol indicating that the next entry is to appear in the actual index file
`encap` (*c*)	`'|'`	Symbol indicating that rest of argument list is to be used as an encapsulating command for the page number
`quote` (*c*)	`'"'`	Symbol that escapes the character following it
`escape` (*c*)	`'\\'`	Symbol without any special meaning unless it is followed by the `quote` character, in which case that character loses its special function and both characters will be printed. This is included because \" is the umlaut character in TeX. The two symbols `quote` and `escape` must be distinct.
`page_compositor` (*s*)	`"-"`	Composite page delimiter

(*s*) attribute of type `string`, (*c*) attribute of type `char` (enclose in single or double quotes, respectively)

Table 11.1: Input style parameters for *MakeIndex*

main source. Such a stand-alone index can be useful if the input text of the book is frozen (the page numbers will no longer change), and you only want to reformat the index.

```
% MakeIndex style file mybook.ist
preamble
 "\\documentclass[12pt]{book} \n\n \\begin{document} \n
  \\begin{theindex}\n"
postamble
 "\n\n\\end{theindex} \n \\end{document}\n"
```

Assuming that the raw index commands are in the file `mybook.idx`, then you can call *MakeIndex* specifying the style file's name:

```
makeindex  -s mybook.ist -o mybookind.tex mybook
```

A nondefault output file name is used to avoid clobbering the source output (presumably `mybook.dvi`). If the index is in file `mybook.ind`, then its typeset output will also be in `mybook.dvi`, thus overwriting the `.dvi` file for the main document.

Keyword	*Default Value*	*Description*
Context		
`preamble` (*s*)	`"\\begin{theindex}\n"`	Preamble command preceding the index
`postamble` (*s*)	`"\n\n\\end{theindex}\n"`	Postamble command following the index
Starting Page		
`setpage_prefix` (*s*)	`"\n\\setcounter{page}{"`	Prefix for the command setting the page
`setpage_suffix` (*s*)	`"}\n"`	Suffix for the command setting the page
New Group/Letter		
`group_skip` (*s*)	`"\n\n\\indexspace\n"`	Vertical space inserted before a new group
`heading_prefix` (*s*)	`""`	Prefix for heading of a new letter group
`heading_suffix` (*s*)	`""`	Suffix for heading of a new letter group
`headings_flag` (*n*)	0	A value `flag=0` inserts nothing between the different letter groups; a value `flag>0` (`<0`) includes an uppercase (lowercase) instance of the symbol characterizing the new letter group, prefixed with `heading_prefix` and appending `heading_suffix`
Entry Separators		
`item_0` (*s*)	`"\n\\item "`	Command to be inserted in front of a level 0 entry
`item_1` (*s*)	`"\n \\subitem "`	Ditto for a level 1 entry starting at level ≥ 1
`item_2` (*s*)	`"\n  \\subsubitem "`	Ditto for a level 2 entry starting at level ≥ 2
`item_01` (*s*)	`"\n \\subitem "`	Command before a level 1 entry starting at level 0
`item_12` (*s*)	`"\n  \\subsubitem "`	Ditto for a level 2 entry starting at level 1
`item_x1` (*s*)	`"\n \\subitem "`	Command to be inserted in front of a level 1 entry when the parent level has no page numbers
`item_x2` (*s*)	`"\n  \\subsubitem "`	Ditto for a level 2 entry
Page Delimiters		
`delim_0` (*s*)	`", "`	Delimiter between an entry and the first page number at level 0
`delim_1` (*s*)	`", "`	Ditto at level 1
`delim_2` (*s*)	`", "`	Ditto at level 2
`delim_n` (*s*)	`", "`	Delimiter between different page numbers
`delim_r` (*s*)	`"-"`	Designator for a page range
Page Encapsulators		
`encap_prefix` (*s*)	`"\\"`	Prefix to be used in front of a page encapsulator
`encap_infix` (*s*)	`"{"`	Infix to be used for a page encapsulator
`encap_suffix` (*s*)	`"}"`	Suffix to be used for a page encapsulator
Page Precedence		
`page_precedence` (*s*)	`"rnaRA"`	Page number precedence: `a`, `A` are lower-, uppercase alphabetic; `n` is numeric; `r` and `R` are lower- and uppercase Roman
Line Wrapping		
`line_max` (*n*)	72	Maximum length of an output line
`indent_space` (*s*)	`"\t\t"`	Indentation commands for wrapped lines
`indent_length` (*n*)	16	Length of indentation for wrapped lines

"`\n`" and "`\t`" are a new line and a tab; (*s*) attribute of type `string`; (*n*) attribute of type `number`

Table 11.2: Output style parameters for *MakeIndex*

Moreover, if you want the page numbers for the index to come out correctly, then you can specify the page number where the index has to start (e.g., 181 in the example below).

```
makeindex  -s mybook.ist  -o mybookind.tex -p 181  mybook
```

MakeIndex can also read the LaTeX log file `mybook.log` to find the page number to be used for the index (see the `-p` option described on page 655).

Changing the "special characters"

The next example shows how you can change the interpretation of special characters in the input file. To do so, you must specify the new special characters in a style file (for instance, `myinchar.ist` shown below). Using Table 11.1 on page 660, in the following example we change the `@` character (see page 651) to `=`, the sublevel indicator `!` (see page 650) to `>`, and the quotation character `"` (see page 652) to `!` (the default sublevel indicator).

```
% MakeIndex style file myinchar.ist
actual '='      % = instead of default @
quote '!'       % !                    "
level '>'       % >                    !
```

In Figure 11.5 on the next page, which should be used in conjunction with the `german` option of the babel package, the double quote character (`"`) is used as a shortcut for the umlaut construct `\"`. This shows another feature of the ordering of *MakeIndex*: namely, the constructs `"` and `\"` are considered to be different entries (`Br"ucke` and `Br\"ucke`, `M"adchen` and `M\"adchen`, although in the latter case the key entry was identical, Maedchen). Therefore, it is important to use the same input convention throughout a complete document.

Changing the output format of the index

You can also personalize the output format of the index. The first thing that we could try is to build an index with a nice, big letter between each letter group. This is achieved with the style `myhead.ist`, as shown below (see Table 11.2 on the preceding page for more details) and gives the result shown in Figure 11.6.

```
% MakeIndex style file myhead.ist
heading_prefix "{\\bfseries\\hfil "     % Insert in front of letter
heading_suffix "\\hfil}\\nopagebreak\n" % Append after letter
headings_flag       1                   % Turn on headings (uppercase)
```

```
" sign, 1
= sign, 2
@ sign, 2
Brücke, 5
Brücke, V
Brücke, v
dimensions
    rule
      width, 3
exclamation (!), 4
    Ah!, 5
Mädchen, c
Mädchen, 3
```

```
Page 1:  \index{\texttt{"} sign}
Page 2:  \index{\texttt{@} sign}
Page 2:  \index{\texttt{!=} sign}
Page 3:  \index{Maedchen=M\"{a}dchen}
Page c:  \index{Maedchen=M"adchen}
Page v:  \index{Bruecke=Br"ucke}
Page 5:  \index{Br"ucke}
Page V:  \index{Br\"ucke}
Page 3:  \index{dimensions>rule>width}
Page 4:  \index{exclamation (!!)}
Page 5:  \index{exclamation (!!)>Ah!!}
```

Figure 11.5: Example of the use of special characters with *MakeIndex*

```
     Symbols
@ sign, 2
       B
box, 21
    dimensions of, 33
    parameters, 5
       D
dimensions
    figure, 17
    rule
      height, 12
      width, 3
    table, 9
       F
fonts
    Computer Modern, 21
    PostScript, 5
       R
rule
    depth, 33, 48
    width, 41
```

```
Page 2:   \index{\texttt{"@} sign}
Page 3:   \index{dimensions!rule!width}
Page 5:   \index{box!parameters}
          \index{fonts!PostScript}
Page 9:   \index{dimensions!table}
Page 12:  \index{dimensions!rule!height}
Page 17:  \index{dimensions!figure}
Page 21:  \index{box}
          \index{fonts!Computer Modern}
Page 33:  \index{box!dimensions of}
          \index{rule!depth}
Page 41:  \index{rule!width}
Page 48:  \index{rule!depth}
```

Figure 11.6: Example of customizing the output format of an index

```
@ sign .................... 2
box ...................... 21
    dimensions of .... 33
    parameters ......... 5
dimensions
    figure ............. 17
    rule
      height .......... 12
      width ............ 3
    table ................ 9
fonts
    Computer Modern 21
    PostScript .......... 5
rule
    depth ......... 33, 48
    width ............. 41
```

```
Page 2:   \index{\texttt{"@} sign}
Page 3:   \index{dimensions!rule!width}
Page 5:   \index{box!parameters}
          \index{fonts!PostScript}
Page 9:   \index{dimensions!table}
Page 12:  \index{dimensions!rule!height}
Page 17:  \index{dimensions!figure}
Page 21:  \index{box}
          \index{fonts!Computer Modern}
Page 33:  \index{box!dimensions of}
          \index{rule!depth}
Page 41:  \index{rule!width}
Page 48:  \index{rule!depth}
```

Figure 11.7: Adding leaders to an index

You could go a bit further and right-adjust the page numbers, putting in dots between the entry and the page number to guide the eye, as shown in Figure 11.7. This effect can be achieved by adding the following commands:

```
% MakeIndex style file myright.ist
delim_0   "\\dotfill "
delim_1   "\\dotfill "
delim_2   "\\dotfill "
```

The LaTeX command `\dotfill` can be replaced by fancier commands, but the underlying principle remains the same.

Treating funny page numbers

As described earlier, *MakeIndex* accepts five basic kinds of page numbers: digits, uppercase and lowercase alphabetic, and uppercase and lowercase Roman numerals. You can also build composed page numbers. The separator character for composed page numbers is controlled by the *MakeIndex* keyword `page_compositor`; the default is the hyphen character (-), as noted in Table 11.1 on page 660. The precedence of ordering for the various kinds of page numbers is given by the keyword `page_precedence`; the default is `rnaRA`, as noted in Table 11.2 on page 661.

Let us start with an example involving simple page numbers. Assume the pages with numbers `ii`, `iv`, `1`, `2`, `5`, `a`, `c`, `A`, `C`, `II`, and `IV` contain an `\index` command with the word `style`. With the default `page_precedence` of `rnaRA` this would be typeset in the index as shown below. The `c` and `C` entries are considered to be Roman numerals, rather than alphabetic characters:

style, ii, iv, c, 1, 2, 5, a, II, IV, C, A

This order can be changed by using the page_precedence keyword, as shown in the style file mypages.ist.

```
% MakeIndex style file mypages.ist
page_precedence "rRnaA"
```

Running *MakeIndex* on the same index entries now yields:

style, ii, iv, c, II, IV, C, 1, 2, 5, a, A

The next step you can take is to use composed page numbers in your document. As noted earlier, the default input separator is the hyphen. Suppose you have a reference to the word style on the following (unsorted) series of pages: C-3, 1-1, D-1-1, B-7, F-3-5, 2-2, D-2-3, A-1, B-5, and A-2. After running *MakeIndex*, the following sorted output is obtained:

style, 1-1, 2-2, A-1, A-2, B-5, B-7, C-3, D-1-1, D-2-3, F-3-5

The separator can be changed to, for example, a dot, by using the page_compositor keyword, shown in the style file mypagsep.ist.

```
% MakeIndex style file mypagsep.ist
page_compositor "."
```

Running *MakeIndex* on the same index entries with the "-" replaced by "." now yields the following results:

style, 1.1, 2.2, A.1, A.2, B.5, B.7, C.3, D.1.1, D.2.3, F.3.5

11.2.5 *MakeIndex* pitfalls

The \index command tries to write its argument unmodified to the .idx file whenever possible.[1] This behavior has a number of different consequences. If the index text contains commands, as in \index{\Prog}, the entry is likely wrongly sorted because in main text this entry is sorted under the sort key \Prog (with the special character \ as the starting sort character) regardless of the definition of the \Prog command. On the other hand, if it is used in some argument of another command, \Prog will expand before it is written to the .idx file; the placement in the index will then depend on the expansion of \Prog. The same thing happens

[1]The way LaTeX deals with the problem of preventing expansion is not always successful. The index package (see Section 11.4.3) uses a different approach that prevents expansion in *all* cases.

when you use `\index` inside your own definitions. That is, all commands inside the index argument are expanded (except when they are robust or preceded by `\protect`).

For sorting, *MakeIndex* assumes that pages numbered with lowercase Roman numerals precede those numbered with Arabic numerals, which in turn precede those numbered with the lowercase alphabet, uppercase Roman numerals and finally the uppercase alphabet. This precedence order can be changed (see the entry `page_precedence` in Table 11.2 on page 661).

MakeIndex will place symbols (i.e., patterns starting with a non-alphanumeric character) before numbers, and before alphabetic entries in the output. Symbols are sorted according to their ASCII values. For word sorting, uppercase and lowercase are considered the same, but for identical words, the uppercase variant will precede the lowercase one. Numbers are sorted in numeric order.

Spaces are treated as ordinary characters when alphabetizing the entries and for deciding whether two entries are the same (see also the example on page 650). Thus, if "␣" denotes a space character, the commands `\index{cat}`, `\index{␣cat}`, and `\index{cat␣}` will produce three separate entries. All three entries look similar when printed. Likewise, `\index{a␣space}` and `\index{a␣␣space}` produce two different entries that look the same on output. For this reason it is important to check for spurious spaces by being careful when splitting the argument of an `\index` command across lines in the input file. The *MakeIndex* option `-c` turns off that behavior and trims leading and trailing white space, compressing all white space within to one blank. We recommend that you use it all the time.

11.3 xindy—An alternative to *MakeIndex*

The xindy program by Roger Kehr and Joachim Schrod is a flexible indexing system that represents an alternative to *MakeIndex*. It avoids several limits, especially for generating indexes in non-English languages. Usage of xindy is recommended in the following cases:

- You have an **index with non-English words** and you want to use a drop-in replacement.
  Migration from *MakeIndex* is easy because xindy can be used without changing the index entries in your document. A compatibility style file will produce results corresponding to *MakeIndex*'s default set-up. The main difference will be that sorting index entries will work out of the box.
- You want to ensure that the index is more **consistent** than that created with *MakeIndex*.
  Because *MakeIndex* takes every indexed term literally, you need to specify index visualization explicitly, as explained in Section 11.1.4 on page 651. In particular, this step is needed if your visualization needs LaTeX commands. If you forget your special visualization in one place, you will get an inconsistent

index. The xindy program takes common LaTeX representations and computes the index key from them—therefore you do not have to specify the difference between the index key and the visualization, every time. (For example, you no longer need the different definitions of `\Index` and `\Indextt` from Section 11.1.7 on page 653.) Of course, you can still provide specific visualizations in your index entry.

- You want more **checks for correctness**.
  If you have an index cross-reference with `see`, as explained in Section 11.1.3 on page 651, xindy checks that the referenced index entry really exists. This way you can avoid dangling references in your indexes.

- You want to create a **technical index in an efficient way**.
  Many technical indexes involve heavy LaTeX markup in the index keys. The xindy program allows user-defined construction of the index keys from this markup. This gives you the ability to emit index entries automatically from your LaTeX commands, so as to get every usage of a technical term into the index. However, you will have to invest the time to define your index key construction rules.

- You want to create an **index with "unusual" terms**.
  For certain terms, special sorting rules exist due to historical reasons. For example, village and people's names are sometimes sorted differently than they're spelled—"St. Martin" is sorted as "Martin" or as "Saint Martin" dependent on context, "van Beethoven" is sorted as "Beethoven", and so on. Symbol indexes are another example where sort order is more or less arbitrarily defined, but should be consistent over a series of work.

The xindy program offers these advantages because it has dropped many of *MakeIndex*'s hard-wired assumptions that are not valid in international documents with arbitrary location reference structures. Instead, xindy provides a flexible framework for configuring index creation, together with a simple *MakeIndex*-like script for standard tasks.

The power of xindy is largely derived from five key features:

Internationalization xindy can be easily configured for languages with different letter sets and/or different sorting rules. You can define extra letters or complete alphabets, and you can provide a set of rules to sort and group them. At the moment, about 50 predefined language sets are available.

Modular configuration xindy is configured with declarations that can be combined and reused. For standard indexing tasks, LaTeX users do not have to do much except grab available modules.

Markup normalization A tedious problem related to technical or multilanguage indexes concerns markup and nontext material. The xindy program allows you to ignore different encodings for the same subject, or to easily strip markup items such as math mode.

User-definable location references An index entry points to locations. Fancy indexes may use not only page numbers, but also book names, law paragraphs, and structured article numbers (e.g., "I-20", "Genesis 1, 31"). The xindy program enables you to sort and group your location references arbitrarily.

Highly configurable markup xindy provides total markup control. This feature is usually not of importance for LaTeX users, but comes in handy for indexing non-TeX material.

Availability If the xindy program is not part of your TeX distribution, its web site (www.xindy.org) offers distributions for many operating systems and more reference documentation. Note that its Windows support is not as good as its UN*X or Linux support. CTAN holds xindy distribution files as well.

11.3.1 Generating the formatted index with xindy

The xindy program comes with a command texindy that allows it to be used in a simple, *MakeIndex*-like way for standard tasks. Options equivalent to those of *MakeIndex* are not described here in detail again; refer to Section 11.2.2 instead. The options -M and -L are described in more detail in the following sections.

texindy [-gilqr] [-o *ind*] [-t *log*] [-L *language*] [-M *module*] [*idx0 idx1* ...]

-i Use standard input (stdin) as the input file.

-o *ind* Take *ind* as the output index file.

-t *log* Take *log* as the transcript file.

-q Operate in quiet mode.

-g Use German mode (equivalent to -L german-din -M german-sty).

-l Use letter ordering; the default is word ordering (equivalent to -M letter-order).

-r Disable implicit page range formation (equivalent to -M no-ranges).

-M *module* Use the xindy module *module* to configure processing.

-L *language* Take *language* as the language configuration for word ordering.

The files *idx0*, *idx1*, and so on contain raw index entries. If you specify more than one input file, you might want to use -o to name the output file, as the default output file name is always computed from *idx0*.

When you use option -c, -p, or -s, you will be warned that these *MakeIndex* options are not supported. In fact, xindy style files are self-written modules and are specified with option -M; Section 11.3.4 explains their creation in more detail.

The texindy command compresses blanks by default, since the authors think that this is the behavior you would expect from an index processor. In fact, the whole TeX program suite works by default under the assumption that sequences of white space are essentially one blank. If you insist on *MakeIndex*-compatible behavior, you can use the module `keep-blanks`, as explained in Section 11.3.3.

MakeIndex has the `-p` option to output a LaTeX command to the `.ind` file that sets the page counter. It may even try to parse the LaTeX log file for that purpose. The xindy program has no such option, and this omission is by design. The xindy authors believe that having a separate LaTeX document for an index is too prone to error and that the ability to include a LaTeX file with the `\printindex` command into the main document is a much better approach.

Indexing LaTeX commands

The texindy command ignores unknown TeX commands by default under the assumption that they do not produce text. It also knows about typical text-producing commands like `\LaTeX` and `\BibTeX` and handles them correctly. If you have your own command definition that produces text, or if you use one supplied by a package, then the entry is sorted incorrectly. You will either need to specify an explicit sort key in your index entry, as in `\index{prog@\Prog}`, or write a xindy style file with a merge rule, as explained in Section 11.3.4.

Be aware that producing index entries in arguments of commands has its own pitfalls, e.g., in `\command{Properties of \Prog\index{\Prog}}`. Then LaTeX commands might be expanded before they are written to the `.idx` file and the placement in the index will depend on the expansion of `\Prog`.

11.3.2 International indexing with xindy

Most non-English languages present additional challenges for index processing. They have accented characters or language-specific characters that obey special rules on how to sort them. It is usually not enough to ignore the accents, and, of course, one must not use the binary encoding of national characters for sorting. In fact, it would be very hard to use binary encoding for sorting even if one wants to—most implementations of LaTeX output many non-ASCII characters as `^^xy`, where `xy` is the hex code of the respective character.

The reality is different: either foreign characters are input with macros, or the inputenc package is used. For example, a LaTeX user in Western Europe on a Linux system is likely to add `\usepackage[latin1]{inputenc}` to all her documents (or on recent Linux distributions the option `utf8`), while a Windows user would use the inputenc option `ansinew` or `utf8`. Then, the raw index file suddenly has lots of LaTeX commands in it, since all national and accented characters are output as commands. In *MakeIndex*, the author needs to separately specify sort and print keys for such index entries. This specification may be managed for some entries, but matters become very error prone if it must be done for all entries that have national characters. In addition, creating index entries automatically by LaTeX commands (as recommended in Section 11.1.7) is no longer possible.

Argument to texindy -L *Option*			
`albanian`	`finnish`	`kurdish-bedirxan`	`slovak-small`
`croatian`	`french`	`kurdish-turkish-i`	`slovak-large`
`czech`	`general` *(default)*	`latvian`	`slovenian`
`danish`	`german-din`	`lithuanian`	`spanish-modern`
`dutch`	`german-duden`	`lower-sorbian`	`spanish-traditional`
`dutch-ij-as-ij`	`greek-translit`	`norwegian`	`swedish`
`english`	`hungarian`	`polish`	`turkish`
`esperanto`	`icelandic`	`portuguese`	`upper-sorbian`
`estonian`	`kurdish`	`romanian`	

`general` *is the default language option and provides definitions approximately well suited for Western European languages, without support for any national characters.*

Additional language options are available for xindy, but may not be used easily with texindy.

Table 11.3: Languages supported by texindy

The xindy program deals with this problem. It knows about LaTeX macros for national characters and handles them as needed. It allows you to define new alphabets and their sort order as well as more complex multiphase sort rules to describe the appropriate sorting scheme. You can then address typical real-world requirements, such as the following:

German German recognizes two different sorting schemes to handle umlauts: normally, *ä* is sorted like *ae*, but in phone books or dictionaries, it is sorted like *a*. The first scheme is known as *DIN order*, the second as *Duden order* [44].

Spanish In Spanish, the ligature *ll* is a separate letter group, appearing after *l* and before *m*.

French In French, the first phase of sorting ignores the diacritics, so that *cote*, *côte*, *coté* and *côté* are all sorted alike. In the next phase, within words that differ only in accents, the accented letters are looked at from right to left. Letters with diacritics then follow letters without them. Thus, *cote* and *côte* come first (no accent on the *e*), and then words with *o* come before words with *ô*.

The xindy program provides *language modules* for a growing number of languages. Such a language module defines the alphabet with all national characters, their sort rules, and letter group definitions adapted to that language. In addition, accented characters commonly used within that language are handled correctly. The predefined language modules cover Western and Eastern European languages. Currently, there is no support available for Asian languages.

There are about 50 predefined languages available, 35 of them are readily usable with texindy. They are listed in Table 11.3 on the facing page; you select one of them with the texindy option `-L`. The other predefined languages have non-Latin scripts, their usage is described in the xindy documentation.

You can also build your own xindy language module. The xindy utility make-rules simplifies this procedure if your language fulfills the following criteria:

- Its script system uses an alphabet with letters.
- It has a sort order based on these letters (and on accents).
- No special context backtracking is required for sorting; accents influence only the sort order of the accented letters.

The xindy web site (`www.xindy.org`) has more information about language module creation with or without make-rules. If you create a new one, please contribute it to the xindy project.

11.3.3 Modules for common tasks

Like *MakeIndex*, xindy may be configured by creating a personal style file, as explained in Section 11.3.4. Most users, however, do not need the full power of xindy configuration. They merely want to solve common problems with a predefined set of possible solutions.

To simplify the completion of common tasks, xindy is distributed with a set of modules, listed in Table 11.4 on the next page. They provide standard solutions for sorting, page range building, and layout requirements. If you have no further demands, you can build your international index without a personal style file; you just specify a language option and the modules you want on the texindy command line. If you use the texindy command, you will deal with three categories of modules:

Automatic modules These modules establish a behavior that is conformant to *MakeIndex*. You cannot turn them off as long as you use the texindy command. If you do not want their behavior, you have to use xindy directly as described in Section 11.3.4.

Default modules Some modules are activated by default and can be turned off with texindy options.

Add-on modules You can select one or more additional modules with the xindy option `-M`.

The automatic module `latex-loc-fmts` indicates a difference between xindy and *MakeIndex*. In *MakeIndex*, you can use a general encapsulation notation to enclose your page number with an arbitrary command (see Section 11.1.4). In xindy,

xindy Module	*Category*	*Description*
Sorting		
`word-order`	Default	A space comes before any letter in the alphabet: "index style" is listed before "indexing"; thus prefix words are listed first. Turn it off with the texindy option `-l`.
`letter-order`	Add-on	Spaces are ignored: "index style" is listed after "indexing". Turn it on with the texindy option `-l`.
`keep-blanks`	Add-on	Leading and trailing white space (blanks and tabs) are not ignored; intermediate white space is not changed.
`ignore-hyphen`	Add-on	Hyphens are ignored: "so-called" is sorted as "socalled".
`ignore-punctuation`	Add-on	All kinds of punctuation characters are ignored: hyphens, periods, commas, slashes, parentheses, and so on.
`numeric-sort`	Auto	Numbers are sorted numerically, not like characters: "V64" appears before "V128".
Page Numbers		
`page-ranges`	Default	Appearances on more than two consecutive pages are listed as a range: "1-4". Turn it off with `-r`.
`ff-ranges`	Add-on	Uses implicit "ff" notation for ranges of three pages, and explicit ranges thereafter: 2f, 2ff, 2-6.
`ff-ranges-only`	Add-on	Uses only implicit ranges: 2f, 2ff.
`book-order`	Add-on	Sorts page numbers with common book numbering scheme correctly—Roman numerals first, then Arabic numbers, then others: i, 1, A-1.
Markup and Layout		
`tex`	Auto	Handles basic TeX conventions.
`latex-loc-fmts`	Auto	Provides LaTeX formatting commands for page number encapsulation.
`latex`	Auto	Handles LaTeX conventions, both in raw index entries and output markup; implies `tex`.
`makeindex`	Auto	Emulates the default *MakeIndex* input syntax and quoting behavior.
`latin-lettergroups`	Auto	Layout contains a single Latin letter above each group of words starting with the same letter.
`german-sty`	Add-on	Handles umlaut markup of babel's `german` and `ngerman` options.

When two entries are identical except for ignored characters, those characters are not ignored any more.

Table 11.4: xindy standard modules

you have to define a location reference class with a corresponding markup definition for each command (see page 678). The `latex-loc-fmts` module provides such definitions for the most common encapsulations, `textbf` and `textit`.

11.3.4 Style files for individual solutions

The xindy program is a highly configurable tool. The chosen functionality is specified in a style file. The texindy command provides convenient access for most purposes, by building a virtual style file from existing modules. If you want to extend the features provided, change functionality, or build your own indexing scheme, you have to use xindy directly and write your own style file, which is just another module. The available xindy modules may be reused.

This section demonstrates how to use xindy with your own style file. It describes the basic concepts underlying the xindy program and gives examples for typical extensions.

The xindy style files are also the means by which you create indexes for non-LaTeX documents (e.g., XML documents, other Unicode-based markup systems). Features used for that purpose are not described in this section as they are beyond the scope of this book. If you're interested, you'll find more material at the xindy web site. To understand xindy style files, we need to present more detail on the basic model that xindy uses. Figure 11.8 on the following page shows the processing steps. A xindy style file contains merge rules, sort rules, location specifications, and markup specifications. Using these declarations, it defines how the raw index from the `.idx` file is transformed into the tagged index in the `.ind` file.

The xindy process model

- Merge rules specify how a sort key is computed from a raw key. The sort key identifies an index entry uniquely—there are no special characters in it. Some index entries come with an explicit sort key, when the `\index{`*sort-key*`@`*raw-key*`}` notation is used. Remember that raw keys may have LaTeX commands in them, or the same index entry may be input in different ways. While some of these differences are created on the author level and are therefore document-specific, most differences are due to expansion of LaTeX commands and thus computation of a sort key can be automated.
- Sort rules declare alphabets, and order within alphabets. The alphabet may not only consist of single characters, but sometimes multiple characters may form a unit for sorting (e.g., `ll` in Spanish). Such new characters must be ordered relative to other characters. A xindy language module consists of alphabet declarations, sort rules, and letter group definitions.
- After sorting, index entries with the same sort key are combined into a consolidated index entry with several locations and a print key. From the raw keys, the first one that appeared in the document is selected as the print key. Ordering, grouping, mixing, and omitting locations to get the final list of locations is a complex task that may be influenced in many ways by location specifications.
- Markup specifications describe which LaTeX commands are added to the consolidated index entries, thus producing a tagged index that can be used as input for LaTeX.

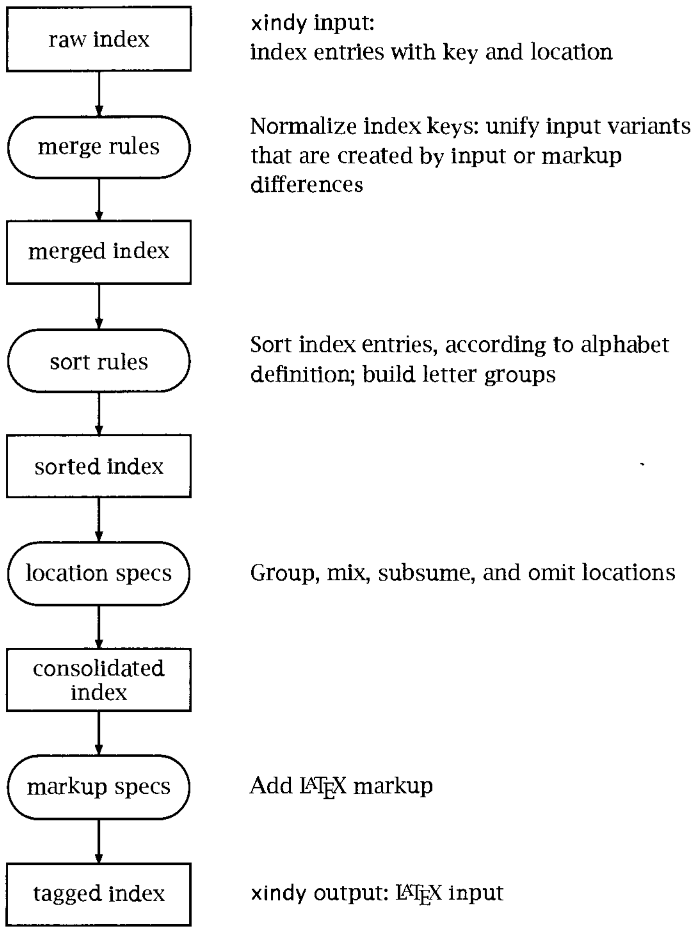


Figure 11.8: xindy process model

Calling **xindy directly**

The xindy options are very similar to those available with texindy. You specify your style file like any other module.

xindy [-qvV] [-o *ind*] [-d *magic*] [-t *log*] [-L *lang*] [-C *codepage*]
[-M *module*] [*idx0 idx1* ...]

-o *ind* Take *ind* as the output index file.

-M *module* Use the xindy module *module* to configure processing.

-L *lang* Take *lang* as the language configuration for word ordering.

-C *codepage* Use *codepage* as internal base encoding for sorting. This is used for fine-grained control of language module selection, needed only for non-Latin scripts.

-q Operate in quiet mode.

-v Operate in verbose mode.

-V Output the version number and terminate.

-d *magic* Produce debugging messages; *magic* decides which xindy component will output them.

Building a xindy style file

A xindy style file will usually start with loading predefined modules that provide much of the desired functionality. Recall that you also have to name explicitly those modules listed as *automatic* (auto) in Table 11.4 on page 672. Afterwards, you can provide definitions of your own that extend or override the already loaded modules.

```
;;; xindy example style file
;; Use this clause for all texindy predefined languages.
(require "tex/inputenc/latin.xdy")

;; merge rules, for markup normalization
;; double backslash needed because it's a regexp
(merge-rule "\\PS *" "Postscript")

;; use texindy automatic modules
(require "texindy.xdy")
;; need to specify default and add-on modules
(require "page-ranges.xdy")  (require "book-order.xdy")

;; markup change: separate page list entries by LaTeX command
(markup-location-list :sep "\page ")
```

Style file syntax

The previous example of a xindy style file showed some of the syntax elements that are available. We now give more precise definitions:

- Basically, a style file consists of a list of declarative clauses in parentheses, starting with a declaration name and followed by several parameters.
- A parameter may be either a string or an option. An option has a keyword, written as :opt, and may have an argument, usually a string but also a number or a fixed value like none. As the name indicates, options are optional; which options are valid depends on the function. A parameter may also comprise a list of parameters in parentheses, as shown in some examples below.
- Comments start with a semicolon and go until the end of line. The examples show a typical way to use different numbers of semicolons: one for inline

comments (after xindy clauses), two for block comments in front of code, and three for comments with "section headers" for the style file. But this is merely a convention—in all places the first semicolon starts the comment.

- Strings are enclosed in double quotes. Newlines are allowed in strings. Within strings, the tilde is an escape character that makes the following letter do something special. For example, ~n specifies a newline.

Merge and sort rules

Merge rules help to normalize raw index entries before sorting and grouping take place. They can be used to unify different notations and to strip the entry from markup material that is irrelevant to sorting. If you merge different index entries, they will appear as one entry and consequently have the same printed representation; that is, all of them will look like the first one that appears in your document. Note that you can only merge single words, not whole phrases.

A merge rule takes two parameters, and declares that occurrences of the first parameter within a word are substituted by the second parameter. Within the second parameter, the virtual characters ~b and ~e may be used: ~b is ordered in front of all other characters, whereas ~e comes after all characters. These two virtual characters are not output, as merge rules are used to construct the sort key from the raw key—and sort keys are internal entry identificators.

Unify index entries

For example, in a city index, places with St in their name may also be written with Saint. Those different spellings should be unified to one index entry nevertheless. In other words, indexing St Barth and Saint Barth shall result in only one index entry.

```
( merge-rule "St"  "Saint" )
```

Unify using regular expressions

In a merge rule, you can also specify a pattern (regular expression) and a replacement string. So-called *extended regexps* are the default and are defined in the POSIX 1003.2 standard. On UN*X systems, you will find their description in the man page of egrep. You can also use *basic regular expressions*, with the option :bregexp in the merge rule. The replacement string may refer to subexpressions, which leads to powerful specifications that are often hard to create and debug. Note also that usage of regular expressions will slow processing down. To index XML tags without angles, you can write:

```
( merge-rule "<(.*)>" "\1" )
```

This will cause \index{<HTML>} and \index{HTML} to be unified as one entry, which may not be the desired effect. To list them separately, but next to each other, you could modify <HTML> to HTML~e as follows:

```
( merge-rule "<(.*)>" "\1~e" )
```

Sort rules specify how characters or character sequences are sorted (i.e., at which position in the alphabet they should be placed). A sort rule consists of two strings. The first string is sorted like the second one. The second string may use `~b` and `~e` to specify the sort order, as explained above.

Letter groups

The xindy program checks for each letter group to see whether it matches a prefix of the entries' sort key. The longest match assigns the index entry to this letter group. If no match is found, the index entry is put into the group `default`.

The following definitions add all entries with the given prefixes to the same letter group *ABC*: *Combine letter groups*

```
( define-letter-group "ABC" :prefixes ("a") )
( define-letter-group "ABC" :prefixes ("b") )
( define-letter-group "ABC" :prefixes ("c") )
```

With indexes that are a bit unbalanced on, say, the letter *X*, you may want to build an extra letter group named *xsl* that contains all entries that start with `xsl:`. These entries will be sorted before all other entries that start with x. *Extra letter groups*

```
( define-letter-group "xsl" :before "x" :prefixes ("xsl:") )
```

Locations

The list of references behind an index entry may contain several groups that have a nonobvious but required order—perhaps Roman numbers, then Arabic numbers, then letters-Arabic numbers combined. We associate this scheme with a typical book having preface matter, normal content, and appendices. In xindy, each such group is called a location class. Within each location class, references are ordered as well. References may be combined to ranges like *10–15* or *5ff.* As you see, xindy allows you to manipulate sorting and range building in various ways.

As an example, to change the minimal length of page ranges, just modify your location class for pages: *Page range length*

```
( define-location-class "pages"
                        ("arabic-numbers")
                        :min-range-length 4 )
```

To suppress ranges for Roman numbers, change the `:min-range-length` option as follows: *Suppress page ranges*

```
( define-location-class "pages"
                        ("roman-numerals-lowercase")
                        :min-range-length none )
```

Nonstandard locations If your raw index contains references with non-numeric components and an unusual syntax (e.g., `Pasta::II.4`), you have to define a special alphabet so that xindy knows how to sort. Use it to define a location class that describes the reference syntax, including separators:

```
( define-alphabet "my-chapters" ("Starters" "Pasta" "Meat" "Sweets") )
( define-location-class "my-index"
                        ("my-chapters" :sep "::"
                         "roman-numerals-uppercase" :sep "."
                         "arabic-numbers") )
```

Location formatting

The xindy program has a very flexible mechanism for formatting, sorting, and grouping locations with special meanings. In your document, you mark up index entries for special formatting, such as `\index{`*keyword* | *definition*`}`. In xindy, you define an attribute with a corresponding markup definition.

You can also configure how your different index entry categories should interact: mix them or list them separately, allow subsuming ranges between them or not, omit entries once part of a range or not.

The following examples illustrate different variations of handling references with special formatting.

Input:	1 4 5 6 **7** 7 **9** 10
Example 1: mix, subsume, omit	1 4-7 **9** 10
Example 2: mix, subsume	1 4-7 **7** **9** 10
Example 3: don't mix, definitions first	**7** **9** 1 4-7 10

Example 1: Mix, subsume, and omit locations.

```
;; mix definition and default
   (define-attributes (("definition" "default")))

;; allow subsuming ranges, omit definition references within ranges
   (merge-to "definition" "default" :drop)

;; define markup
   (markup-location :attr "definition" :open "\textbf{" :close "}" )
```

Example 2: Mix and subsume locations.

```
;; mix definition and default
   (define-attributes (("definition" "default")))

;; allow subsuming ranges, keep definition references within ranges
   (merge-to "definition" "default")
```

```
;; define markup
   (markup-location :attr "definition" :open "\textbf{" :close "}" )
```

Example 3: Do not mix locations, list definitions first.

```
;; separate definition and default, definitions come first
   (define-attributes (("definition") ("default")))

;; define markup
   (markup-location :attr "definition" :open "\textbf{" :close "}" )
```

Note that `:define-attributes` has one parameter in parentheses. It consists of either one list of attribute names enclosed in parentheses or a list of strings, each string enclosed in parentheses. All attributes that are together in one brace are mixed. If you have several attributes, an expression like

```
(("definition" "important") ("default"))
```

would indicate that definitions may be mixed with the group of important references, but not with default references.

11.4 Enhancing the index with LaTeX features

This section describes LaTeX's support for index creation. It presents possibilities to modify the index layout and to produce multiple indexes.

11.4.1 Modifying the layout

You can redefine the environment `theindex`, which by default is used to print the index. The layout of the `theindex` environment and the definition of the `\item`, `\subitem`, and `\subsubitem` commands are defined in the class files article, book, and report. In the book class you can find the following definitions:

```
\newenvironment{theindex}
   {\@restonecoltrue\if@twocolumn\@restonecolfalse\fi
    \twocolumn[\@makeschapterhead{\indexname}]%
    \@mkboth{\MakeUppercase\indexname}{\MakeUppercase\indexname}%
    \thispagestyle{plain}\parindent\z@  \parskip\z@ \@plus .3\p@\relax
    \columnseprule \z@  \columnsep 35\p@   \let\item\@idxitem}
   {\if@restonecol\onecolumn\else\clearpage\fi}
\newcommand\@idxitem  {\par\hangindent 40\p@}
\newcommand\subitem   {\par\hangindent 40\p@ \hspace*{20\p@}}
\newcommand\subsubitem{\par\hangindent 40\p@ \hspace*{30\p@}}
```

Although this is programmed in a fairly low-level internal language, you can probably decipher what it sets up. First it tests for two-column mode and saves the result. Then it sets some spacing parameters, resets the page style to plain, and calls \twocolumn. Finally it changes \item to execute \@idxitem, which produces a paragraph with a hanging indention of 40 points. A higher-level reimplementation (using ifthen) might perhaps look as follows:

```
\renewenvironment{theindex}
   {\ifthenelse{\boolean{@twocolumn}}{\setboolean{@restonecol}{false}}%
                                     {\setboolean{@restonecol}{true}}%
    \twocolumn[\chapter*{\indexname}]%
    \markkboth{\MakeUppercase\indexname}{\MakeUppercase\indexname}%
    \setlength\parindent{0pt}\setlength\parskip{0pt plus 0.3pt}%
    \setlength\columnseprule{0pt}\setlength\columnsep{35pt}%
    \thispagestyle{plain}\let\item\@idxitem }
   {\ifthenelse{\boolean{@restonecol}}{\onecolumn}{\clearpage}}
```

Adjusting this definition allows you to make smaller modifications, such as changing the page style or the column separation.

You can also make an index in three rather than two columns. To do so, you can use the multicol package and the multicols environment:

```
\renewenvironment{theindex}{%
   \begin{multicols}{3}[\chapter*{\indexname}][10\baselineskip]%
   \addcontentsline{toc}{chapter}{\indexname}%
   \setlength\parindent{0pt}\pagestyle{plain}\let\item\@idxitem}
  {\end{multicols}}
```

We require at least 10 lines of free space on the current page; otherwise, we want the index to start on a new page. In addition to generating a title at the top, we enter the heading as a "Chapter" in the table of contents (.toc) and change the page style to plain. Then the \item command is redefined to cope with index entries (see above), and the entries themselves are typeset in three columns using the multicols environment.

11.4.2 showidx, repeatindex, tocbibind, indxcite—Little helpers

Several useful little LaTeX packages exist to support index creation. A selection is listed in this section, but by browsing through the on-line catalogue [169] you will probably find additional ones.

Show index entries in margin

The package showidx (by Leslie Lamport) can help you improve the entries in the index and locate possible problems. It shows all \index commands in the margin of the printed page. Figure 11.4 on page 656 shows the result of including the showidx package.

Handle page breaks gracefully

The package repeatindex (by Harald Harders) repeats the main item of an index if a page or column break occurs within a list of subitems. This helps the reader correctly identify to which main item a subitem belongs.

The package tocbibind (by Peter Wilson) can be used to add the table of contents itself, the bibliography, and the index to the *Table of Contents* listing. See page 48 for more information on this package. *Table of contents support*

The package indxcite (by James Ashton) automatically generates an author index based on citations made using BibTeX. This type of functionality is also available with the bibliography packages natbib and jurabib, both of which are described in detail in Chapter 12. *Automatic author index*

11.4.3 index—Producing multiple indexes

The index package (written by David Jones and distributed as part of the camel package) augments LaTeX's indexing mechanism in several areas:

- Multiple indexes are supported.
- A two-stage process is used for creating the raw index files (such as the default `.idx` file) similar to that used to create the `.toc` file. First the index entries are written to the `.aux` file, and then they are copied to the `.idx` file at the end of the run. With this approach, if you have a large document consisting of several included files (using the `\include` command), you no longer lose the index if you format only part of the document with `\includeonly`. Note, however, that this makes the creation of a chapter index more difficult.
- A starred form of the `\index` command is introduced. In addition to entering its argument in the index, it typesets the argument in the running text.
- To simplify typing, the `\shortindexingon` command activates a shorthand notation. Now you can type `^{foo}` for `\index{foo}` and `_{foo}` for `\index*{foo}`. These shorthand notations are turned off with the `\shortindexingoff` command. Because the underscore and circumflex characters have special meanings inside math mode, this shorthand notation is unavailable there.
- The package includes the functionality of the showidx package. The command `\proofmodetrue` enables the printing of index entries in the margins. You can customize the size and style of the font used in the margin with the `\indexproofstyle` command, which takes a font definition as its argument (e.g., `\indexproofstyle{\footnotesize\itshape}`).
- The argument of `\index` is never expanded when the index package is used. In standard LaTeX, using `\index{\command}` will sometimes write the expansion of `\command` to the `.idx` file (see Section 11.2.5 on page 665). With the index package, `\command` itself is always written to the `.idx` file. While this is helpful in most cases, macro authors can be bitten by this behavior. In Section 11.1.7, we recommended that you define commands that automatically add index entries. Such commands often expect that `\index` will expand its parameter and they may not work when you use the index package. Be careful and check the results of the automatic indexing—this is best practice, anyhow.

You can declare new indexes with the `\newindex` command. The command `\renewindex`, which has an identical syntax, is used to redefine existing indexes.

`\newindex{`*tag*`}{`*raw-ext*`}{`*proc-ext*`}{`*indextitle*`}`

The first argument, *tag*, is a short identifier used to refer to the index. In particular, the commands `\index` and `\printindex` are redefined to take an optional argument—namely, the tag of the index to which you are referring. If this optional argument is absent, the index with the tag "default" is used, which corresponds to the usual index. The second argument, *raw-ext*, is the extension of the raw index file to which LaTeX should write the unprocessed entries for this index (for the default index it is `.idx`). The third argument, *proc-ext*, is the extension of the index file in which LaTeX expects to find the processed index (for the default index it is `.ind`). The fourth argument, *indextitle*, is the title that LaTeX will print at the beginning of the index.

As an example we show the set-up used to produce this book. The preamble included the following setting:

```
\RequirePackage{index}
\proofmodetrue              % while proofing the index entries
\newindex{xauthor}{adx}{and}{People}
\newindex{xcmds}{cdx}{cnd}{Index of Commands and Concepts}
```

In the backmatter, printing of the index was done with the following lines:

```
\printindex[xcmds] \printindex[xauthor]
```

For each generated raw index file (e.g., `tlc2.adx` for the list of authors) we ran *MakeIndex* to produce the corresponding formatted index file for LaTeX:

```
makeindex -o tlc2.and -t tlc2.alg tlc2.adx
```

While all of these tools help to get the correct page numbers in the index, the real difficulty persists: choosing useful index entries for your readers. This problem you still have to solve (if you are lucky, with help).

In fact, the index of this book was created by a professional indexer, Richard Evans of Infodex Indexing Services in Raleigh, North Carolina. Dick worked closely with Frank to produce a comprehensive index that helps you, the reader, find not only the names of things (packages, programs, commands, and so on) but also the tasks, concepts, and ideas described in the book. But let him tell you (from the Infodex FAQ at `http://www.mindspring.com/~infodex`):

Question: Why do I need an indexer? Can't the computer create an index?

Answer: To exactly the same degree that a word processor can write the book. Indexes are creative works, requiring human intellect and analysis.

LaTeX can process the indexing markup, but only a human indexer can decide what needs to be marked up. Our sincere thanks to Dick for his excellent work.

CHAPTER 12

Managing Citations

12.1 Introduction

Citations are cross-references to bibliographical information outside the current document, such as to publications containing further information on a subject and source information about used quotations. It is certainly not necessary to back everything by a reference, but background information for controversial statements, acknowledgments of other work, and source information for used material should be given.

There are numerous ways to compile bibliographies and reference lists. They can be prepared manually, if necessary, but usually they are automatically generated from a database containing bibliographic information (see Chapter 13). This chapter introduces some of the many presentation forms of bibliographical sources and it reviews different traditions regarding how such sources are referred to in a document.

The chapter begins with a short introduction to the major citation schemes in common use. This is followed by a description of LaTeX's standard markup for bibliography lists and its interface to the BibTeX program that can be used to produce such lists automatically from a (suitably prepared) document source. More detailed information on BibTeX is then given in Chapter 13. In the current chapter we are only interested in how BibTeX can be used to produce a bibliography list.

Armed with this knowledge we plunge into a detailed discussion of how LaTeX supports the different citation schemes. At the time we wrote the first edition of this book, LaTeX basically supported the "number-only" system. A decade later, the situation has changed radically. Today, most major citation schemes are well supported by extension packages.

We end this chapter by discussing packages that can deal with multiple bibliographies in one document. This is not difficult if the reference lists are prepared manually, but it poses some challenges if you want to interact with BibTeX, as well.

12.1.1 Bibliographical reference schemes

There are four common methods of referring to sources: the "short-title", "author-date", "author-number", and "number-only" systems. The first of these is often used in books on humanities; the second appears mainly in science and social science works. The other two are less often used, although the last is quite common within the LaTeX world, as it has been actively promoted by Leslie Lamport and originally was the only form of citation supported by LaTeX. Outside the LaTeX world a variation of it, called "numeric by first citation", is quite popular as well.

The short-title system

In the short-title system, the reference to a source is given directly in the text, either inline or as a footnote, often in the form "Hart, *Hart's Rules*, p. 52". In the context of the publication, if abbreviations for the title are established, the form "Goossens et al., *LGC*" may appear as an alternative. Many variations exist. For instance, the first time a work is cited it might be presented with a lot of detail; later references might then use a shorter form—citing only the author's name and a short title or the year. In case of repeated citations to the same work in direct succession, you might find *Ibid.* instead of a repeated reference. An implementation of the short-title system that allows all kinds of customizations is provided by the **jurabib** package (see Section 12.5.1).

Because in the short-title system a full reference is usually given the first time a work is cited, you can omit a list of references or a bibliography that contains all cited works in a single place.

The author-date system

In the author-date system (often referred to as the Harvard system after one of its better known typographical variants), references to sources are also given directly in the text. This time, however, they show the author's name (or names) and the year of the publication. The full citation is given in a list of references or a bibliography. If the author published more than one work in a given year, that year is suffixed with lowercase letters (e.g., 2001a, 2001b).

There have been many attempts over the years to provide author-date citation support for LaTeX. With the **natbib** package (discussed in Section 12.3.2) there is now a very flexible and general solution available.

In all citation schemes that use author names, a work by three or more authors is usually referred to by using the name of the first author followed by *et al.* Especially with the author-date system, this may lead to ambiguous citations if different groups containing the same main author published in the same year. This problem can be seen in the following example.

```
\usepackage{chicago} \bibliographystyle{chicago}
Entries with multiple authors can be problematical, e.g., \shortcite{LGC97}
and \shortcite{test97} or worse \shortcite{LGC97,test97}. \bibliography{tex}
```

Entries with multiple authors can be problematical, e.g., (Goossens et al. 1997) and (Goossens et al. 1997) or worse (Goossens et al. 1997; Goossens et al. 1997).

References

Goossens, M., S. Rahtz, and F. Mittelbach (1997). *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript.* Tools and Techniques for Computer Typesetting. Reading, MA, USA: Addison-Wesley Longman.

Goossens, M., B. User, J. Doe, et al. (1997). Ambiguous citations. Submitted to the IBM Journal of Research and Development.

12-1-1

In the above example the bibliography is produced from the sample BibTeX database `tex.bib` shown in Figure 12.2 on page 690. This database is used in most examples in this chapter. Above we applied the BibTeX style `chicago` to it, a style that aims to implement a bibliography and reference layout as suggested by *The Chicago Manual of Style* [38].

One way to resolve such ambiguous citations is to use all author names in such a case, although that approach will lead to lengthy citations and is not feasible if the number of authors exceeds a certain limit. Another solution is to append a, b, and so on, to the year, even though the citations are actually for different author groups. This strategy is, for example, advocated in [29]. If the bibliography is compiled manually, as outlined in Section 12.1.2, this result can be easily achieved. When using BibTeX, you have to use a BibTeX style file that recognizes these cases and provides the right data automatically. For example, the style `chicago` cannot be used in this case, but all BibTeX styles produced with makebst (see Section 13.5.2) offer this feature:

Entries with multiple authors might be problematical, e.g., Goossens et al. [1997a] and Goossens et al. [1997b] or even Goossens et al. [1997a,b]. But then they might not.

References

M. Goossens, S. Rahtz, and F. Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript.* Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997a. ISBN 0-201-85469-4.

M. Goossens, B. User, J. Doe, et al. Ambiguous citations. Submitted to the IBM J. Res. Dev., 1997b.

12-1-2

```
\usepackage{natbib}
\bibliographystyle
              {abbrvnat}
Entries with multiple
authors might
be problematical,
e.g., \cite{LGC97} and
\cite{test97} or even
\cite{LGC97,test97}.
But then they might not.

\bibliography{tex}
```

The author-number system

In the author-number system, the references to the sources are given in the form of the author's name (or names) followed by a number, usually in parentheses or brackets, indicating which publication of the author is cited. In the corresponding bibliography all publications are numbered on a per-author (or author group) basis. In the LaTeX world this system is fairly uncommon as it is difficult to produce manually. As far as we know, there is currently no BibTeX support available for it, though this situation might change in the future. A variation of the above is to number all publications sequentially. For this case suitable BibTeX styles exist.

The number-only system

Finally, in the number-only system, publications are sequentially numbered in the bibliography. Citations in the text refer to these numbers, which are usually surrounded by brackets or parentheses. Sometimes raised numbers are used instead. In a slight variation, known as "alpha" style, citations comprise the author's name and the year of the publication. Thus, the bibliographic label and the citation may look like "[Knu86]".

One argument against this system—put forward, for example, in *The Chicago Manual of Style* [38]—is that it raises the costs of publication since a late addition or deletion of a reference may require renumbering and consequently costly (and error-prone) changes to many pages throughout the manuscript. With automatic cross-referencing facilities as provided by LaTeX, this argument no longer holds true. In fact, the number-only system is the default system provided with LaTeX.

Numerical by first citation

A fairly popular form of the number-only system numbers the publications sequentially by their first citation in the text (and presents them in that order in the bibliography). This is fairly easy to provide with LaTeX. The next two sections and Section 12.2.3 explain how to avoid references in the table of contents that might mess up the expected order.

12.1.2 Markup structure for citations and bibliography

The standard LaTeX environment for generating a list of references or a bibliography is called `thebibliography`. In its default implementation it automatically generates an appropriate heading and implements a vertical list structure in which every publication is represented as a separate item.

```
\begin{thebibliography}{widest-label}
\bibitem[label1]{cite-key1} bibliographic information
\bibitem[label2]{cite-key2} bibliographic information
    ...
\end{thebibliography}
```

The *widest-label* argument is used to determine the right amount of indentation for individual items. If the works are numbered sequentially, for example, it should contain the number of items.

Individual publications are introduced with a `\bibitem` command. Its mandatory argument is a unique cross-reference *key* that refers to this publication in the text. The optional argument defines the textual representation that is used in the citation and as the *label* in the list. If this argument is not specified, the publications are numbered with Arabic numerals by default. Within a publication the command `\newblock` may be used to separate major blocks of information.

Depending on the layout produced by the class, it may result in a normal space, some extra space, or in starting a new line.

References

[1] Goossens, M., S. Rahtz, and F. Mittelbach (1997). *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript.* Reading, MA, USA: Addison-Wesley Longman.

[2] Goossens, M., B. User, J. Doe (1997). Ambiguous citations.

12-1-3

```
\begin{thebibliography}{2}
\bibitem{LGC97} Goossens, M., S.~Rahtz,
   and F.~Mittelbach (1997).
  \newblock \emph{The \LaTeX{} Graphics Companion:
  Illustrating Documents with \TeX{} and
  PostScript}. \newblock Reading, MA, USA:
  Ad\-di\-son-Wes\-ley Longman.
\bibitem{GUD97} Goossens, M., B.~User,  J.~Doe
  (1997). \newblock Ambiguous citations.
\end{thebibliography}
```

Producing a large bibliography manually in this way is clearly a tedious and difficult task and the result is normally not reusable, as nearly all journals and publishers have their own house styles with different formatting requirements. For this reason it is generally better to use BibTeX, a program that generates ready-to-use LaTeX input from a database of bibliographical information. This is discussed in the next section.

Note that without the optional argument to `\bibitem` the references are numbered in the order in which they appear in the bibliography. Thus, if you produce the bibliography manually, numbering and sorting them by order of first citation becomes your task. In contrast, when using BibTeX, this result can be achieved automatically.

Order by first citation done manually

Inside a document, publications are cited by referring to the *cite-key* arguments of the `\bibitem` commands. For this purpose LaTeX offers the `\cite` command, which takes such a key as its argument. It can, in fact, take a comma-separated list of such keys and offers an optional argument to specify additional information such as page or chapter numbers. The precise syntax is described in Section 12.2.1. For short-title or author-date citation schemes, additional citation commands are available once the supporting packages are loaded.

12.1.3 Using BibTeX to produce the bibliography input

The BibTeX program gathers all citation keys used in a document, looks them up in a bibliographical database, and generates a complete `thebibliography` environment that can be loaded by LaTeX in a subsequent run. Depending on the BibTeX style used, it can either sort the entries according to some scheme (e.g., author names, year of publication) or produce a bibliography with entries in the order in which they appear in the `.aux` file. Note that using such a "nonsorting" style automatically generates a bibliography by order of first citation as required by the house styles of many publishers. An example of such a BibTeX style is `unsrt`.

Order by first citation produced with BibTeX

The procedure for running LaTeX and BibTeX is shown schematically in Figure 12.1 on the next page. At least three LaTeX runs are necessary—first to produce

① Run LaTeX, which generates a list of `\cite` references in its auxiliary file, `.aux`.

② Run BibTeX, which reads the auxiliary file, looks up the references in a database (one or more `.bib` files), and then writes a file (the `.bbl` file) containing the formatted references according to the format specified in the style file (the `.bst` file). Warning and error messages are written to the log file (the `.blg` file). Note that BibTeX never reads the original LaTeX source file.

③ Run LaTeX again, which now reads the `.bbl` file containing the bibliographic information.

④ Run LaTeX a third time, resolving all references.

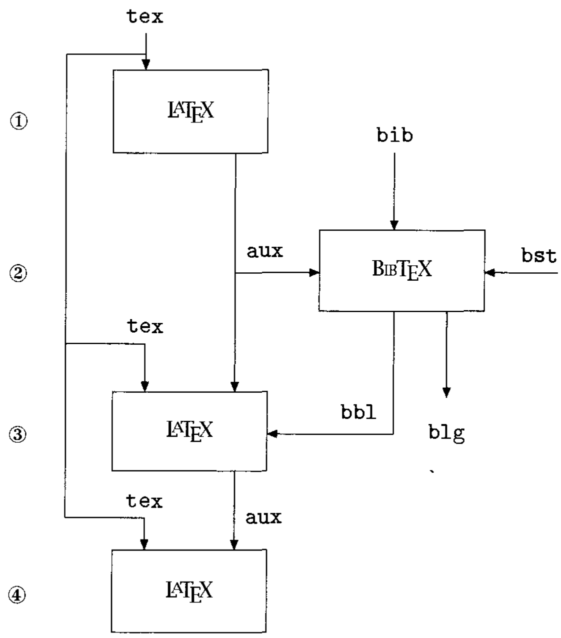


Figure 12.1: Data flow when running BibTeX and LaTeX

data for BibTeX, then to load the result from the BibTeX run, and finally to resolve the cross-references to the bibliographical list added by the previous run.

`\bibliography{`*file-list*`}` `\bibliographystyle{`*style*`}`

To inform BibTeX which databases are to be searched to resolve citations, you should specify their names, separated by commas (and without the extension `.bib`), as an argument to the command `\bibliography`. This command should be placed at the point where the bibliography should finally appear. In addition, you have to tell BibTeX how the bibliographic entries should be formatted. This is done by using the command `\bibliographystyle` in the preamble with a suitable BibTeX style as its argument. It is, of course, important that the *cite-keys* used in the document uniquely identify an entry in the database file(s), so that the citation reference can be resolved when the document is processed.

To enable BibTeX to access the information without the need to parse the LaTeX source files, these commands write two lines to the `.aux` file. For a similar reason the `\cite` command, as well as any variant of it, writes its *key* to this file. For example, in Example 12-1-2 the `.aux` file would contain (beside other entries):

```
\bibstyle{abbrvnat}
\citation{LGC97}
```

```
\citation{test97}
\bibdata{tex}
```

Do not confuse these commands with those intended for use in the document. They exist solely to facilitate internal communication between LaTeX and BibTeX. If you mistakenly use `\bibdata` instead of `\bibliography`, then LaTeX will process your document without failure, but BibTeX will complain that it does not find any database information in the `.aux` file.

The precise format of a BibTeX entry will be described in detail in Chapter 13. To be able to understand the examples in the next sections more easily, you should nonetheless know that the basic structure of a BibTeX entry consists of three parts:

1. A publication *entry type* (e.g., "`book`", "`article`", "`inproceedings`", "`phdthesis`").
2. A *user-chosen keyword* identifying the publication. If you want to reference the entry in your document, then the argument *cite-key* of the `\cite` command should be identical (also in case) to this keyword.
3. A *series of fields* consisting of a field identifier with its data between quotes or curly braces (e.g., "`author`", "`journal`", and "`title`").

A sample database is shown in Figure 12.2 on the following page. This database is used in most examples throughout the chapter to show how applying different BibTeX style files to it results in different presentation forms.

Various schemes exist for conveniently associating bibliography keywords with their entries in a database. A popular one is the so-called Harvard system, where you take the author's surname (converted to lowercase) and the year of publication, and combine them using a colon (e.g., `smith:1987`).

BibTeX entries are read by BibTeX in the bibliography database (the `.bib` file), and the formatting of the entries is controlled by an associated bibliography style (the `.bst` file), which contains a set of instructions written in a stack-based language. The latter is interpreted by the BibTeX program (see Section 13.6).

BibTeX knows which fields are required, optional, and ignored for any given entry type (see Table 13.1 on page 763). It will issue warnings, such as "`author name required`", if something is missing. The style file can control the typesetting of both the citation string in the main text and the actual bibliography entry inside the `thebibliography` environment.

It is important to remember that BibTeX is not required for managing citations (except for the package jurabib and those packages intended for producing multiple bibliographies). You can produce a bibliography without BibTeX by providing the bibliographic entries yourself using the syntax described in Section 12.1.2. It is also a simple matter to manually edit the output from BibTeX to cope with special cases. Moreover, if your LaTeX document has to be self-contained, you can include the contents of the `.bbl` file in your document.

```
@String{ttct    = "Tools and Techniques for Computer
                   Typesetting"   }

@Book{LGC97,
  author =        "Michel Goossens and Sebastian Rahtz
                   and Frank Mittelbach",
  title =         "The {\LaTeX} Graphics Companion:
                   Illustrating Documents with {\TeX}
                   and {PostScript}",
  publisher =     "Ad{\-d}i{\-s}on-Wes{\-l}ey Longman",
  address =       "Reading, MA, USA",
  pages =         "xxi + 554",
  year =          "1997",
  ISBN =          "0-201-85469-4",
  series =        ttct
}
@UNPUBLISHED{test97,
 author = "Michel Goossens and Ben User
           and Joe Doe and others",
 title  = "Ambiguous citations",
 year   = "1997",
 note   = "Submitted to the " # ibmjrd
}
@Book{LWC99,
  author =        "Michel Goossens and Sebastian Rahtz",
  title =         "The {\LaTeX} {Web} companion:
                   integrating {\TeX}, {HTML},
                   and {XML}",
  publisher =     "Ad{\-d}i{\-s}on-Wes{\-l}ey Longman",
  address =       "Reading, MA, USA",
  pages =         "xxii + 522",
  year =          "1999",
  ISBN =          "0-201-43311-7",
  note =          "With Eitan M. Gurari and Ross Moore
                   and Robert S. Sutor",
  series =        ttct
}
@Book{Knuth-CT-a,
  Author          = "Donald E. Knuth",
  Title           = "The {\TeX}book",
  Publisher       = "Ad{\-d}i{\-s}on-Wes{\-l}ey",
  Address         = "Reading, MA, USA",
  Volume          = "A",
  Series          = "Computers and Typesetting",
  pages           = "ix + 483",
  year            = 1986,
  isbn            = "0-201-13447-0",
}
@Article{Knuth:TB10-1-31,
  Author          = "Donald E. Knuth",
  Title           = "{Typesetting  Concrete
                     Mathematics}",
  Journal         = "TUGboat",
  Volume          = "10",
  Number          = "1",
  Pages           = "31--36",
  year            = 1989,
  month           = apr,
  issn            = "0896-3207"
}
```

```
@Book{vLeunen:92,
  author       = "Mary-Claire van Leunen",
  gender       = "sf",
  title        = "A handbook for scholars",
  publisher    = "Oxford University Press",
  address      = "Walton Street, Oxford OX2 6DP, UK",
  pages        = "xi + 348",
  year         = "92"
}

@manual{GNUMake,     key = {make},
 title = {{GNU Make}, A Program for Directing
 Recompilation},  organization= "Free
 Software Foundation",address = "Boston,
 Massachusetts",ISBN={1-882114-80-9},year = 2000}

@book{G-G,
 TITLE     = {{Gutenberg Jahrbuch}},
 EDITOR    = {Hans-Joachim Koppitz},
 PUBLISHER = {Gutenberg-Gesellschaft, Internationale
               Vereinigung f\"ur Geschichte und
               Gegenwart der Druckkunst e.V.},
 ADDRESS   = {Mainz, Germany},
 NOTE      = {Contains results on the past and present
             history of the art of printing. Founded
             by Aloys Ruppel. Published since 1926.}
}
@misc{oddity,
  title        = "{{TUGboat} The Communications of the
                    {\TeX} User Group}",
  howpublished = "Quarterly published.",
  year         = {1980ff},
 }
@InProceedings{MR-PQ,
 author     = "Frank Mittelbach and Chris Rowley",
 title      = "The Pursuit of Quality: How can
               Automated Typesetting achieve the
               Highest Standards of Craft
               Typography?",
 pages      = "261--273",
 crossref   = "EP92"}
@InProceedings{Southall,
 Author        = "Richard Southall",
 Title         = "Presentation Rules and Rules of
                  Composition in the Formatting of
                  Complex Text",
 Pages         = "275--290",
 crossref      = "EP92"}
@Proceedings{EP92,
 title       = "{EP92}---Proceedings of Electronic
                Publishing, '92",
 shorttitle  = "{EP92}",
 editor      = "Christine Vanoirbeek and Giovanni Coray",
 publisher   = "Cambridge University Press",
 address     = "Cambridge",
 year        = 1992,
 booktitle   = "{EP92}---Proceedings of Electronic
                Publishing, '92"
}
```

Figure 12.2: Sample BIBTEX database `tex.bib`

This database uses different conventions in individual entries (e.g., lower-, upper-, or mixed-case field names, different indentations) to show some features and problems in later examples. By applying one of the tools from Section 13.4 it could be normalized.

12.2 The number-only system

12.2.1 Standard LaTeX—Reference by number

As mentioned earlier in this chapter, the number-only system is the default citation method directly supported by standard LaTeX. That is, without loading any additional packages, it is the only method supported by the provided markup commands. Bibliographic citations inside the text of a LaTeX document are then flagged with the command `\cite`.

`\cite[`*text*`]{`*key*`}`   `\cite[`*text*`]{`*key1*`,`*key2*`,...}`   `\nocite{`*key-list*`}`

The `\cite` command associates each keyword in the list in its mandatory argument with the argument of a `\bibitem` command from the `thebibliography` environment to produce the citation reference. As with other LaTeX identifiers, these keys are case-sensitive.

The citation numbers generated are defined by the order in which the keys appear on the `\bibitem` commands inside the `thebibliography` environment or, if an optional argument is used with `\bibitem`, by the data provided in that argument.

The optional parameter *text* is an additional note, which will be printed together with the text generated by the `\cite` command as shown in the following example. For comparison we have used an unbreakable space (~) in the first citation and a small space (`\,`) in the second. Of course, such typographical details should be handled uniformly throughout a publication.

Color support for LaTeX is described in [2, chap. 9] and the `hyperref` package in [1, pp. 35–67].

12-2-1

```
\bibliographystyle{plain}
Color support for \LaTeX{} is described in
\cite[chap.~9]{LGC97} and the \texttt{hyperref}
package in \cite[pp.\,35--67]{LWC99}.
```

To save space, the examples in this chapter often omit the bibliography list. They are generated by placing `\bibliography{tex}` at the end of the example document when automatically generating the example output for the book. Thus, you should read examples such as 12-2-1 as follows: the result is produced by generating the bibliography with BibTeX, applying the style `plain` (shown), and using the database `tex.bib` (not shown; see Figure 12.2). Thus, the actual document that produced the example contained `\bibliography{tex}` near the end.

A note on the examples in this chapter

In conjunction with BibTeX, you can use the `\nocite` variant of the `\cite` command. Its sole purpose is to write the keys from the *key-list* argument into the `.aux` file, so that the associated bibliography information will appear in the bibliography even if the publication is otherwise not cited. For technical reasons it has to appear *after* `\begin{document}`, even though it does not produce any output and would logically be best placed in the preamble. It can be used as often as necessary. As a special case `\nocite{*}` includes all entries of the chosen BibTeX data in the list of references.

As stated above, the association between a `\cite` command and one or more bibliography entries is made via the *key-list* argument. The citation text, which will actually appear in the typeset text, depends on the chosen bibliographic style.

Customizing citation references and the bibliography

Unfortunately, standard LaTeX is not equipped with an easily customizable interface through which you can adjust the formatting of the citation references. Thus, to change the default brackets around the numbers into parentheses, for example, we need to redefine the internal LaTeX command `\@cite`.

Even worse, the user-level `\cite` command sets the internal temporary switch `@tempswa` to indicate whether an optional argument was present. Thus, if we want to handle that optional argument, we need to evaluate the value of that switch. The `\@cite` command receives two arguments: the list of obtained references and the note (if present). In the following example we typeset `(#1` and, if `@tempswa` is true, follow it by a comma and ␣#2. This is then followed by the closing parenthesis. The `\nolinebreak[3]` ensures that a break after the comma is taken only reluctantly.

```
\bibliographystyle{plain}     \usepackage{ifthen}
\makeatletter
\renewcommand\@cite[2]{({#1\ifthenelse{\boolean{@tempswa}}
                                     {,\nolinebreak[3] #2}{}})}
\makeatother
Color support for \LaTeX{} is described in \cite{LGC97} and
the \texttt{hyperref} package in \cite[pp.\,35--67]{LWC99}.
```
12-

Color support for LaTeX is described in (2) and the hyperref package in (1, pp. 35–67).

The redefinition of `\@cite` for purposes like the above can be avoided by loading the cite package; see Section 12.2.2.

For the `thebibliography` environment, which holds the list of the actual references, the situation is unfortunately not much better—the default implementation offers few customization possibilities. To modify the layout of the labels in front of each publication (e.g., to omit the brackets), you have to change the internal LaTeX command `\@biblabel`.

```
\bibliographystyle{abbrv}
\makeatletter
\renewcommand\@biblabel[1]{#1.}
\makeatother
\nocite{Knuth-CT-a,Knuth:TB10-1-31}
\bibliography{tex}
```
12-2

References

1. D. E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
2. D. E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, Apr. 1989.

Packages that implement a variation of the author-date system (e.g., the apalike, chicago, or natbib package), typically unconditionally redefine `\@biblabel` to simply swallow its argument and typeset nothing. After all, such a bibliography is used by looking up the author name, so a label is unnecessary. The natbib package is somewhat more careful: if it detects that \@biblabel was changed, then it honors the redefinition.

As mentioned earlier, different blocks of information, such as the authors or the title, are separated inside one \bibitem in the bibliography by \newblock commands, which are also automatically inserted by most BibTeX styles. Normally, bibliographic entries are typeset together in one paragraph. If, however, you want your bibliography to be "open", with each block starting on a new line with succeeding lines inside a block indented by a length \bibindent (default 1.5 em), then the class option openbib should be specified. This option is supported by all standard classes. The result is shown in the next example; we also redefine \@biblabel to get raised labels.

References

[1] M. Goossens and S. Rahtz.
The LaTeX Web companion: integrating TeX, HTML, and XML.
Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1999.
With Eitan M. Gurari and Ross Moore and Robert S. Sutor.

[2] D. E. Knuth.
Typesetting Concrete Mathematics.
TUGboat, 10(1):31–36, Apr. 1989.

12-2-4

```
\documentclass[openbib]{article}
\bibliographystyle{abbrv}
\setlength\bibindent{24pt}

\makeatletter
\renewcommand\@biblabel[1]
            {\textsuperscript{#1}}
\makeatother
\nocite{LWC99,Knuth:TB10-1-31}
\bibliography{tex}
```

12.2.2 cite—Enhanced references by number

One shortcoming that becomes readily apparent when you use LaTeX's default method of citing publications is the fact that it faithfully keeps the order of citations as given in the *key-list* argument of the \cite command. The following example therefore shows a very strangely ordered list of numbers (the unresolved reference was added deliberately):

Good information about TeX and LaTeX can be found in [2, 1, 3, ?, 4].

12-2-5

```
\bibliographystyle{plain}
Good information about \TeX{} and \LaTeX{} can be found in
\cite{LGC97,LWC99,Knuth-CT-a,Knuth:ct-b,Knuth:TB10-1-31}.
```

This situation can be easily improved by simply loading the cite package (by Donald Arseneau), as in the following example:

Good information about TeX and LaTeX can be found in [?, 1–4].

12-2-6

```
\usepackage{cite}      \bibliographystyle{plain}
Good information about \TeX{} and \LaTeX{} can be found in
\cite{LGC97,LWC99,Knuth-CT-a,Knuth:ct-b,Knuth:TB10-1-31}.
```

By default, the cite package sorts citation numbers into ascending order, representing three or more consecutive numbers as a number range. Any non-numeric label is moved to the front (in the above example the "?" generated by the unresolved reference). If sorting is not desired you can globally prevent it by loading the package with the option `nosort`. Compression into ranges can be suppressed by using the option `nocompress`.

Customizing the citation layout

To customize the typeset reference the cite package offers a number of commands. For example, `\citeleft` and `\citeright` determine the material placed on the left and right sides of the citation string, respectively. These commands can be used to typeset parentheses instead of brackets as seen in the following example, which should be compared to Example 12-2-2 on page 692. We can also redefine `\citemid`, the separation between citation and optional note, to produce a semicolon and a space.

Color support for LaTeX is described in (2) and the `hyperref` package in (1; pp. 35–67).

```
\usepackage{cite}     \bibliographystyle{plain}
\renewcommand\citeleft{(} \renewcommand\citeright{)}
\renewcommand\citemid{;\nolinebreak[3] }
Color support for \LaTeX{} is described in \cite{LGC97} and
the \texttt{hyperref} package in \cite[pp.\,35--67]{LWC99}.
```

12-2-7

Customizing breaks within citations

Another important aspect of citation management is controlling the behavior near the end of a line. Consider the string "see [2-3,7,13]". Besides not allowing any kind of line break within this string, one could allow breaking after the "see", after the commas, or after the en dash in a range.

By default, the cite package discourages line breaks before the citation with `\nolinebreak[3]`, discourages line breaks after a comma separating the optional note with `\nolinebreak[2]`, and very strongly discourages line breaks after en dashes in a range and after commas separating individual citation numbers. You can control the last three cases by redefining `\citemid`, `\citedash`, and `\citepunct`. For example, to prevent breaks after the en dashes while allowing breaks after commas without much penalty, you could specify

```
\renewcommand\citedash{\mbox{--}\nolinebreak}
\renewcommand\citemid{,\nolinebreak[1] }
\renewcommand\citepunct{,\nolinebreak[1]\hspace{.13em plus .1em minus .1em}}
```

There are several interesting points to note here. All three definitions are responsible not only for controlling any line breaks but also for adding the necessary punctuation: a dash for the range, a comma and a full blank before the optional note, or a comma and a tiny space between individual citations. For instance, if you want no space at all between citations, you can redefine `\citepunct` to contain only a comma. The other important and probably surprising aspect is the `\mbox` surrounding the en dash. This box is absolutely necessary if you want to control LaTeX's ability to break at this point. TeX automatically adds a break point after an explicit hyphen or dash, so without hiding it in a box, the `\nolinebreak` command would never have any effect—the internally added break point would still allow a line break at this point. Finally, the `\hspace` command allows for some stretching or shrinking; if you prefer a fixed space instead, remove the `plus` and `minus` components.

The high penalty that is added before a citation is hard-wired in the code. It is, however, inserted only if you have not explicitly specified a penalty in your document. For instance, "`see~\cite{..}`" will be honored and no break will happen between "see" and the citation.

Customizing citation numbers

One more customization command, `\citeform`, allows you to manipulate the individual reference numbers. By default, it does nothing, so the labels are typeset unchanged. In the following example we colored them. Other kinds of manipulation are possible, too (e.g., adding parentheses in Example 12-2-9).

12-2-8

Color support for LaTeX is described in [2] and the `hyperref` package in [1, pp. 35–67].

```
\usepackage{cite,color}  \bibliographystyle{plain}
\renewcommand\citeform[1]{\textcolor{blue}{#1}}
Color support for \LaTeX{} is described in \cite{LGC97} and
the \texttt{hyperref} package in \cite[pp.\,35--67]{LWC99}.
```

`\citen{`*key-list*`}`

The package offers an additional command, `\citen` (its aliases are `\citenum` and `\citeonline`), that can be used to get a list of numbers without the surrounding `\citeleft` and `\citeright` (e.g., the default brackets). Other formatting is still done. In the next example we surround individual references to citations with parentheses, something that admittedly looks a little strange when used together with the default bracketing of the whole citation.

12-2-9

(1)–(3),(5) but [(4), §5]

```
\usepackage[nospace]{cite}  \bibliographystyle{plain}
\renewcommand\citeform[1]{(#1)}
\citen{LGC97,LWC99,test97,vLeunen:92} but \cite[\S5]{Knuth-CT-a}
```

The package offers a number of options to handle standard configuration requests or to influence the package behavior in other ways. Some of them have already been discussed, but here is the full list:

`adjust/noadjust` Enables (default) or disables "smart" handling of space before a `\cite` or `\citen` command. By default, spaces before such commands are normalized to an interword space. If you write `see\cite{..}`, a space is inserted automatically.

`compress/nocompress` Enables (default) or disables compression of consecutive numbers into ranges.

`sort/nosort` Enables (default) or prevents sorting of the numbers.

`space` A full interword space is used after commas, and breaking at this point is not actively discouraged. The default (option not specified) is to use a small space and to discourage, but allow, breaking.

`nospace` Eliminates the spaces after commas in the list of numbers, but retains the space after the comma separating the optional note. The result of this option shown in Example 12-2-9 on the previous page. It is not the opposite of the space option!

verbose By default, cite warns only once per reference for undefined citations. When this option is specified, the warning is repeated each time an undefined reference is cited.

Citations with superscript numbers

The latest release of the cite package can also display citation references as superscript numbers if the package is loaded with the option superscript (or super). In the past this ability was provided by the separate package overcite (developed by the same author), which is still available for compatibility reasons.

If the \cite command is used with an optional argument, then the whole list of citations will be typeset as though the cite package was loaded without the superscript option.

With the superscript or super option in effect, the customization commands \citeleft, \citeright, and \citemid affect only citations with an optional argument, while \citedash, \citepunct, and \citeform affect all citations. For details of their use, see the discussion on pages 694–695.

Good information about TeX and LaTeX can be found in.[?,1–4] For hyperref see (1, pp. 35–67).

```
\usepackage[superscript]{cite}  \bibliographystyle{plain}
\usepackage{color}
\renewcommand\citeform[1]{\textcolor{blue}{#1}}
\renewcommand\citeleft{(} \renewcommand\citeright{)}
Good information about \TeX{} and \LaTeX{} can be found in
\cite{LGC97,LWC99,Knuth-CT-a,Knuth:ct-b,Knuth:TB10-1-31}.
For \texttt{hyperref} see \cite[pp.\,35--67]{LWC99}.
```

12-2-10

You will probably not need to change your source document, regardless of whether the superscript option is used. In particular, a space before the citation command will be ignored if the citations are raised. In principle, you can add this option without having to adjust your document sources, provided your writing style does not use the numerical citation as part of the sentence structure, as in the above example.

If superscript numbers are used for citation labels, special care is needed when punctuation characters surround the citation. By default, the cite package automatically moves a punctuation character following a citation in front of the superscript. Punctuation characters that will migrate in this way are stored in the command \CiteMoveChars, with ".,;:" being the default (! and ? are not included, but can be added). A problem that can result from this process is doubling of periods. This case is detected by the package and one punctuation character is suppressed; see the second citation in the next example.

... book;[2] see also Goossens et al.[1]

```
\usepackage[superscript]{cite}     \bibliographystyle{plain}
\ldots\ book~\cite{Knuth-CT-a}; see also Goossens et al.~\cite{LGC97}.
```

12-2-11

Unfortunately, with capitalized abbreviations or the use of \@ *after* a period, the suppression of double periods fails. Possible workarounds are shown in the next example. Note, however, that the solution with `U.S.A\@.` only works together with the cite package, but it gives the wrong spacing if no citation is present (you are effectively claiming that the sentence ends after the abbreviation)!

	et al..[1]	et al.[1]
12-2-12	U.S.A..[?]	U.S.A.[?]

```
\usepackage[super]{cite}    \bibliographystyle{plain}
et al.\@ \cite{LGC97}.  \hfil   et al.\ \cite{LGC97}.\par
U.S.A. \cite{unknown}.  \hfil   U.S.A\@. \cite{unknown}.
```

There is yet another pitfall that you may encounter: the final punctuation character does not migrate inside a preceding quotation—a style, for example, advocated by *The Chicago Manual of Style* [38]. In this case you may have to rewrite part of your source text accordingly.

For details see "The TEXbook".[1] But
12-2-13 wanted is "The TEXbook."[1]

```
\usepackage[super]{cite}    \bibliographystyle{plain}
For details see ``The \TeX book'' \cite{Knuth-CT-a}.
But wanted is ``The \TeX book.'' \cite{Knuth-CT-a}
```

The main options of the cite package were discussed on page 695. Three more options related to raising the reference numbers exist. With the option `nomove` specified, punctuation characters are not migrated before the superscript citation. With the option `ref` specified, citations with an optional argument have the word "Ref." prepended. This is internally implemented by changing `\citeleft`, so if you want a different string or want to change from brackets to, say, parentheses, you have to redefine the customization commands instead of using this option.

Color support is described in "LGC"[2] and the `hyperref` pack-
12-2-14 age in "LWC" [Ref. 1, pp. 35–67].

```
\usepackage[super,ref]{cite}  \bibliographystyle{plain}
Color support is described in ``LGC'' \cite{LGC97} and the
\texttt{hyperref} package in ``LWC'' \cite[pp.\,35--67]{LWC99}.
```

Finally, the `biblabel` option raises the labels in the bibliography. (By default, they retain their default layout regardless of whether you use the option `superscript` or its alias `super`.)

12.2.3 notoccite—Solving a problem with unsorted citations

If you want the publications in the bibliography to appear in exactly the order in which they are cited in the document, then you should use unsorted citation styles (e.g., the BIBTEX style `unsrt`). This approach will not work, however, if citations are present inside headings or float captions. In that case, these citations will also appear in the table of contents or list of figures, and so on. As a result they will be moved to the beginning of the bibliography even though they appear much later in the text.

You can circumvent this problem by specifying an optional argument for `\caption`, `\section`, or similar commands without the citation, so that no citations will be written into such tables. If you have to use citations in these places, then a "manual" solution is to first delete any auxiliary files left over from previous LaTeX runs, then run LaTeX once, and then run BibTeX. In that case BibTeX will pick up only citations from the main document. Clearly, this approach is prone to error and you may find that your citation order got mangled after all when you finally see your article in print.

Donald Arseneau developed the small package notoccite to take care of this problem by redefining the internal command `\@starttoc` in such a way that citations do not generate `\citation` commands for BibTeX within the table of contents and similar lists. Simply loading that package will take care of the problem in all cases—provided you have not used some other package that redefines `\@starttoc` (for example, notoccite cannot be combined with hyperref or the AMS document classes).

12.3 The author-date system

Depending on the structure of the sentence, the author-date system normally uses one of two different forms for references: if the author's name appears naturally in the sentence, it is not repeated within the parentheses or brackets; otherwise, both the author's name and the year of publication are used. This style poses an unsolvable problem when LaTeX's standard syntax should be used, as only one command (`\cite`) is available.

Consequently, anyone developing support for the author-date system has had to extend the LaTeX syntax for citing publications. The following example shows the two forms and their implementation (with two new commands) as provided by the natbib system.

Knuth (1989) shows ... This is explained in the authoritative manual on TeX (Knuth, 1986).

```
\usepackage{natbib}
\citet{Knuth:TB10-1-31} shows \ldots\ This is explained
in the authoritative manual on \TeX{}~\citep{Knuth-CT-a}.
```

12-3-1

Extending the LaTeX syntax for citing publications does not solve the problem completely. In order to produce the different forms of citation references needed in the author-date system, the information that is passed back from the bibliography through the optional argument of the `\bibitem` command needs to be structured. Without a special structure it is impossible to pick up the data needed for the textual references (e.g., producing just the year in parentheses). That is, a bibliographical entry like

```
\bibitem[Donald~E. Knuth 1986]{Knuth-CT-a}    Donald~E. Knuth.
  \newblock \emph{The {\TeX}book}, volume~A of \emph{Computers and
    Typesetting}. \newblock Addison-Wesley, Reading, MA, USA, 1986.
```

will allow the `\cite` command to produce "(Donald E. Knuth 1986)" but not "Donald E. Knuth (1986)" or just "Knuth" or just "1986" as well. You also have to ensure that \bibitem does not display the label, but that outcome can be fairly easily arranged.

The solution used by all implementations for author-date support is to introduce a special syntax within the optional argument of \bibitem. In some implementations this structure is fairly simple. For instance, chicago requires only

```
\bibitem[\protect\citeauthoryear{Goossens, Rahtz, and Mittelbach}
                                {Goossens et~al.}{1997}]{LGC97}
```

This information can still be produced manually, if needed. Other packages go much further and encode a lot of information explicitly. For example, jurabib asks for the following kind of argument structure (same publication):

```
\bibitem[{Goossens\jbbfsasep Rahtz\jbbstasep Mittelbach\jbdy {1997}}%
 {}{{0}{}{book}{1997}{}{}{}{xxi + 554}{Reading, MA, USA\bpubaddr {}
 Ad{\-d}i{\-s}on-Wes{\-l}ey Longman\bibbdsep {} 1997}}{{The {\LaTeX}
 Graphics Companion: Illustrating Documents with {\TeX} and {PostScript}}%
 {}{}{}{}{}{}{}{}}]{LGC97}
```

As we shall see (Section 12.5.1), this approach gives a lot of flexibility when referring to the publication, but it is clear that no one wants to produce a bibliography environment with such a structure manually. Hence, the only usable solution in this case is to use an external tool like BIBTEX to generate the entries automatically.

12.3.1 Early attempts

Over the years several independent add-on packages have been developed to support the author-date system. Unfortunately, each one introduced a different set of user-level commands. Typically, the add-ons consist of a LATEX package providing the user commands and one or more BIBTEX styles to generate the thebibliography environment with a matching syntax in the optional argument of the \bibitem command.

For example, the chicago package, which aimed to implement the recommendations of *The Chicago Manual of Style* [38], offers the following list of commands (plus variants all ending in NP to omit the parentheses—for example, \citeNP):

(Goossens, Rahtz, and Mittelbach 1997)
(Goossens, Rahtz, and Mittelbach)
Goossens, Rahtz, and Mittelbach (1997)
(Goossens and Rahtz 1999)
(Goossens and Rahtz)
Goossens and Rahtz (1999)
12-3-2 (1999), 1999

```
\usepackage{chicago} \bibliographystyle{chicago}
\cite{LGC97}                \\
\citeA{LGC97}               \\
\citeN{LGC97}               \\
\shortcite{LWC99}           \\
\shortciteA{LWC99}          \\
\shortciteN{LWC99}          \\
\citeyear{LWC99}, \citeyearNP{LWC99}
```

Several BIBTEX styles (chicago, chicagoa, jas99, named, and newapa) are compatible with the chicago package. All of them are still in use, even though the package itself is rarely included in LATEX distributions these days (natbib can be used instead to provide the user-level syntax).

In contrast, only two commands are provided by David Rhead's authordate1-4 package, the original support package for the BIBTEX styles authordate1 to authordate4. It implements recommendations by the Cambridge and Oxford University Presses and various British standards.

(Goossens *et al.* , 1997) or (1997)

```
\usepackage{authordate1-4}
\bibliographystyle{authordate2}
\cite{LGC97}  or  \shortcite{LGC97}
```

12-3-3

As a final example we look briefly at the harvard package by Peter Williams and Thorsten Schnier. In contrast to the two previously described packages, harvard has been further developed and updated for LATEX 2ε. It implements a number of interesting features. For example, a first citation gives a full author list, whereas a later citation uses an abbreviated list (unless explicitly requested otherwise). The user-level commands are shown in the next example.

(Goossens, Rahtz & Mittelbach 1997)
(Goossens et al. 1997) second citation
(Goossens, Rahtz & Mittelbach 1997) long names forced
Goossens et al. (1997)
(e.g., Goossens et al. 1997)
Goossens et al.
Knuth's (1986)

```
\usepackage{harvard}
\bibliographystyle{agsm}
\cite{LGC97}                               \\
\cite{LGC97} \hfill second citation  \\
\cite*{LGC97}\hfill long names forced\\
\citeasnoun{LGC97}                       \\
\citeaffixed{LGC97}{e.g.,}              \\
\citename{LGC97}                          \\
\possessivecite{Knuth-CT-a}
```

12-3-4

The harvard package requires a specially prepared bibliography environment in which `\bibitem` is replaced by `\harvarditem`, a command with a special syntax used to carry the information needed for author-date citations. A few BIBTEX styles (including agsm, dcu, kluwer, and nederlands) implement this special syntax.

Many of these packages support the author-date system quite well. Nevertheless, with different packages using their own syntax and supporting only half a dozen BIBTEX styles each, the situation stayed unsatisfactory for a long time. Matters changed for the better when Patrick Daly published his natbib support package, described in the next section.

12.3.2 natbib—Customizable author-date references

Although most publishers will indicate which bibliographic style they prefer, it is not always evident how to change from one system to the other if one has to prepare source texts adhering to multiple styles.

To solve the problem of incompatible syntaxes described in the previous section, Patrick Daly developed the natbib package (for "NATural sciences BIBliography"). This package can accept several `\bibitem` variants (including `\harvarditem`) as produced by the different BibTeX styles. Thus, for the first time, (nearly) all of the author-date BibTeX styles could be used with a single user-level syntax for the citation commands.

The natbib package is compatible with packages like babel, chapterbib, hyperref, index, and showkeys, and with various document classes including the standard LaTeX classes, amsbook and amsart, classes from the KOMA-Script bundle, and memoir. It cannot be used together with the cite package, but provides similar sorting and compressing functions via options.

The natbib package therefore acts as a single, flexible interface for most of the available bibliographic styles when the author-date system is required. It can also be used to produce numerical references, as we will see in Section 12.4.1.

The basic syntax

The two central commands of natbib are `\citet` (for textual citation) and `\citep` (for parenthetical citation).

`\citet[`*post-note*`]{`*key-list*`}`	`\citet[`*pre-note*`][`*post-note*`]{`*key-list*`}`
`\citep[`*post-note*`]{`*key-list*`}`	`\citep[`*pre-note*`][`*post-note*`]{`*key-list*`}`

Both commands take one mandatory argument (the *key-list* that refers to one or more publications) and one or two optional arguments to add text before and after the citation. LaTeX's standard `\cite` command can take only a single optional argument denoting a *post-note*. For this reason the commands implement the following syntax: with only one optional argument specified, this argument denotes the *post-note* (i.e., a note placed after the citation); with two optional arguments specified, the first denotes a *pre-note* and the second a *post-note*. To get only a *pre-note* you have to add an empty second argument, as seen in lines 4 and 8 in the next example. Also note that natbib redefines `\cite` to act like `\citet`.[1]

Goossens et al. (1997)
Goossens et al. (1997, chap. 2)
Goossens et al. (see 1997, chap. 2)
pre-note only: Goossens et al. (see 1997)

(Goossens et al., 1997)
(Goossens et al., 1997, chap. 2)
(see Goossens et al., 1997, chap. 2)
12-3-5 pre-note only: (see Goossens et al., 1997)

```
\usepackage{natbib}
\citet{LGC97}                            \\
\citet[chap.~2]{LGC97}                   \\
\citet[see][chap.~2]{LGC97}              \\
pre-note only: \citet[see][]{LGC97}      \\[5pt]
\citep{LGC97}                            \\
\citep[chap.~2]{LGC97}                   \\
\citep[see][chap.~2]{LGC97}              \\
pre-note only: \citep[see][]{LGC97}
```

[1] To be precise, `\cite` is redefined to act like `\citet` if natbib is used in author-date mode as discussed in this section. If used in author-number mode (see Section 12.4.1), it works like `\citep`.

Both commands have starred versions, \citet* and \citep* (with otherwise identical syntax), that will print the full list of authors if it is known.[1] These versions will work only when this feature is supported by the used BIBTEX style file. In other words, the information must be made available through the optional argument of \bibitem; if it is missing, the abbreviated list is always printed.

Goossens, Rahtz, and Mittelbach (1997)
(see Goossens, Rahtz, and Mittelbach, 1997)

```
\usepackage{natbib}
\citet*{LGC97}                 \\
\citep*[see][]{LGC97}
```

12-3-6

Two other variant forms exist: \citealt works like \citet but does not generate parentheses, and \citealp is \citep without parentheses. Evidently, some of the typeset results come out almost identically.

Goossens et al. 1997
Goossens et al., 1997
Goossens, Rahtz, and Mittelbach 1997
Goossens, Rahtz, and Mittelbach, 1997
Goossens and Rahtz, 1999, p. 236 etc.

```
\usepackage{natbib}
\citealt{LGC97}                \\
\citealp{LGC97}                \\
\citealt*{LGC97}               \\
\citealp*{LGC97}               \\
\citealp[p.~236]{LWC99} etc.
```

12-3-7

When using the author-date system it is sometimes desirable to just cite the author(s) or the year. For this purpose natbib provides the following additional commands (\citeauthor* is the same as \citeauthor when the full author information is unavailable):

Goossens et al.
Goossens, Rahtz, and Mittelbach
1997 or (1997)

```
\usepackage{natbib}
\citeauthor{LGC97}             \\
\citeauthor*{LGC97}            \\
\citeyear{LGC97}  or  \citeyearpar{LGC97}
```

12-3-8

Even more complex mixtures of text and citation information can be handled with the command \citetext. It takes one mandatory argument and surrounds it with the parentheses used by other citation commands. By combining this command with \citealp or other commands that do not produce parentheses, all sorts of combinations become possible.

(see Goossens et al., 1997 or Knuth, 1986)

```
\usepackage{natbib}
\citetext{see \citealp{LGC97} or \citealp{Knuth-CT-a}}
```

12-3-9

Forcing names to upper case

Sometimes a sentence starts with a citation, but the (first) author of the cited publication has a name that starts with a lowercase letter. In that case the commands discussed so far cannot be used. The natbib package solves this problem by providing for all commands variants that capitalize the first letter. They are

[1] If you plan to also use the jurabib package (see Section 12.5.1), then avoid the starred forms as they are not supported by that package.

easy to remember: just capitalize the first letter of the corresponding original command. For example, instead of \citet*, use \Citet*. Here are some additional examples.

Normal citation: van Leunen (92)
Van Leunen (92) or Van Leunen 92
(Van Leunen, 92) or Van Leunen, 92
Van Leunen

12-3-10

```
\usepackage{natbib}
Normal citation: \citet{vLeunen:92}              \\
\Citet{vLeunen:92} or \Citealt{vLeunen:92}  \\
\Citep{vLeunen:92} or \Citealp{vLeunen:92}  \\
\Citeauthor{vLeunen:92}
```

As a final goody, natbib lets you define alternative text for a citation that can be used instead of the usual author-date combination. For the definition use \defcitealias (usually in the preamble), and for the retrieval use \citetalias or \citepalias.

Goossens et al. (1997) = Dogbook II
(Goossens et al., 1997) = (Dogbook II)
Alias changed: (see Dogbook II 2ed)

12-3-11

```
\usepackage{natbib} \defcitealias{LGC97}{Dogbook~II}
\citet{LGC97} = \citetalias{LGC97}  \\
\citep{LGC97} = \citepalias{LGC97}  \par
\defcitealias{LGC97}{Dogbook~II~2ed}
Alias changed: \citepalias[see][]{LGC97}
```

With the commands introduced in this section, natbib offers the same features (with minor differences) as other support packages for the author-date system (e.g., the packages described in Section 12.3.1). In addition, it provides features not found elsewhere. On the other hand, in a few cases natbib does not offer directly equivalent commands. For example, harvard's \possessivecite command (shown in Example 12-3-4) has no direct correspondence in natbib, but it can be easily built manually. To emulate it, you can either directly use \citeauthor and \citeyearpar, as is done in the first line of the next example, or define your own command if this type of construction is used more often.

Knuth's (1986)
Knuth's (1986)

12-3-12

```
\usepackage{natbib} \bibliographystyle{agsm}
\newcommand\possessivecite[1]{\citeauthor{#1}'s \citeyearpar{#1}}
\citeauthor{Knuth-CT-a}'s \citeyearpar{Knuth-CT-a} \\
\possessivecite{Knuth-CT-a}
```

Multiple citations

In standard LaTeX, multiple citations can be made by including more than one citation *key-list* argument to the \cite command. The same is possible for the citation commands \citet and \citep (as well as their variant forms). The natbib package then automatically checks whether adjacent citations in the *key-list* have the same author designation. If so, it prints the author names only once. This feature requires that the author names be spelled identically. For instance, natbib will consider "D. Knuth" and "Donald Knuth" to be two different authors.

Goossens et al. (1997); Goossens and Rahtz (1999)
(Goossens et al., 1997; Goossens and Rahtz, 1999)
(Knuth, 1989, 1986)

```
\usepackage{natbib}
\citet{LGC97,LWC99}      \\
\citep{LGC97,LWC99}      \\
\citep{Knuth:TB10-1-31,Knuth-CT-a}
```

12-3-13

The last line in the previous example exhibits a potential problem when using several keys in one citation command: the references are typeset in the order of the *key-list*. If you specify the option `sort`, then the citations are sorted into the order in which they appear in the bibliography, usually alphabetical by author and then by year.

(Knuth, 1986, 1989)

```
\usepackage[sort]{natbib}
\citep{Knuth:TB10-1-31,Knuth-CT-a}
```

12-3-14

While all the citation commands support *key-lists* with more than one citation key, they are best confined to `\citep`; already `\citet` gives questionable results. The situation gets worse when you use optional arguments: with `\citet` any *pre-note* is added before each year (which could be considered a defect in the package). More generally, it is not at all clear what these notes are supposed to refer to. Hence, if you want to add notes it is better to separate your citations.

(see van Leunen, 92; Knuth, 1986, p. 55)
(see Knuth, 1986, 1989, p. 55)
van Leunen (see 92); Knuth (see 1986, p. 55)
Knuth (see 1986, 1989, p. 55)

```
\usepackage{natbib}
\citep[see][p.~55]{vLeunen:92,Knuth-CT-a}  \\
\citep[see][p.~55]{Knuth-CT-a,Knuth:TB10-1-31} \\
\citet[see][p.~55]{vLeunen:92,Knuth-CT-a}  \\
\citet[see][p.~55]{Knuth-CT-a,Knuth:TB10-1-31}
```

12-3-15

Full author list only with the first citation

The harvard package automatically typesets the first citation of a publication with the full list of authors and subsequent citations with an abbreviated list. This style of citation is quite popular in some disciplines, and natbib supports it if you load it with the option `longnamesfirst`. Compare the next example to Example 12-3-4 on page 700.

(Goossens, Rahtz & Mittelbach 1997) first citation
(Goossens et al. 1997) second
(Goossens, Rahtz & Mittelbach 1997) names forced
Goossens et al. (1997)
(e.g., Goossens et al. 1997)
Goossens et al.

```
\usepackage[longnamesfirst]{natbib}
\bibliographystyle{agsm}
\citep{LGC97} \hfill first citation  \\
\citep{LGC97} \hfill         second  \\
\citep*{LGC97}\hfill   names forced  \\
\citet{LGC97}                        \\
\citep[e.g.,][]{LGC97}               \\
\citeauthor{LGC97}
```

12-3-16

Some BIBTEX style files are quite cleverly programmed. For example, when the `agsm` BIBTEX style, used in the previous example, detects that shortening a list of authors leads to ambiguous citations, it will refuse to produce an abbreviated list. Thus, after adding the `test97` citation to the example, all citations suddenly come out in long form.[1] BIBTEX styles produced with makebst avoid such ambiguous citations by adding a suffix to the year, but other BIBTEX styles (e.g., `chicago`) happily produce them; see Example 12-3-18 below.

(Goossens, Rahtz & Mittelbach 1997)	first citation
(Goossens, Rahtz & Mittelbach 1997)	second
(Goossens, User, Doe et al. 1997)	first citation
(Goossens, User, Doe et al. 1997)	second citation

12-3-17

```
\usepackage[longnamesfirst]{natbib}
\bibliographystyle{agsm}
\citep{LGC97} \hfill  first citation  \\
\citep{LGC97} \hfill          second  \\
\citep{test97}\hfill  first citation  \\
\citep{test97}\hfill second citation
```

Some publications have so many authors that you may want to always cite them using their abbreviated name list, even the first time. You can achieve this effect by listing their keys, separated by commas, in the argument of the `\shortcites` declaration. This example also shows that use of the `chicago` style can lead to ambiguous citations (lines 1 and 2 versus line 5).

(Goossens et al., 1997)	first citation
(Goossens et al., 1997)	second citation
(Goossens, Rahtz, and Mittelbach, 1997)	forced
(Goossens, User, Doe, et al., 1997)	first citation
(Goossens et al., 1997)	second citation

12-3-18

```
\usepackage[longnamesfirst]{natbib}
\bibliographystyle{chicago}
\shortcites{LGC97}
\citep{LGC97} \hfill  first citation \\
\citep{LGC97} \hfill second citation \\
\citep*{LGC97}\hfill          forced \\
\citep{test97}\hfill  first citation \\
\citep{test97}\hfill second citation
```

Customizing the citation reference layout

So far, all of the examples have shown round parentheses around the citations, but this is by no means the only possibility offered by natbib. The package internally knows about more than 20 BIBTEX styles. If any such style is chosen with a `\bibliographystyle` command, then a layout appropriate for this style is selected as well. For example, when using the agu style (American Geophysics Union) we get:

Goossens et al. [1997]
[*Knuth*, 1986; *Goossens and Rahtz*, 1999]
12-3-19 [see *Knuth*, 1986, chap. 2]

```
\usepackage{natbib} \bibliographystyle{agu}
\citet{LGC97}  \\  \citep{Knuth-CT-a,LWC99} \\
\citep[see][chap.~2]{Knuth-CT-a}
```

By default, the citation layout is determined by the chosen BIBTEX style (or natbib's defaults if a given style is unknown to natbib). By including a `\citestyle` declaration you can request to use the citation style associated with a BIBTEX style that is different from the one used to format the bibliography. In the next example

[1] Something that puzzled the author when he first encountered it while preparing the examples.

we use the agsm style for the citations while the overall style remains agu. If you compare this example to Example 12-3-19 you see that the textual formatting is unchanged (e.g., italic for author names), but the parentheses and the separation between authors and year have both changed.

Goossens et al. (1997)
(*Knuth* 1986, *Goossens and Rahtz* 1999)
(see *Knuth* 1986, chap. 2)

```
\usepackage{natbib} \bibliographystyle{agu}
\citestyle{agsm}
\citet{LGC97} \\ \citep{Knuth-CT-a,LWC99} \\
\citep[see][chap.~2]{Knuth-CT-a}
```

12-3-20

It is also possible to influence the layout by supplying options: round (default for most styles), square, curly, or angle will change the type of parentheses used, while colon[1] (default for most styles) and comma will change the separation between multiple citations. In the next example, we overwrite the defaults set by the agu style, by loading natbib with two options.

Goossens et al. {1997}
{*Knuth*, 1986, *Goossens and Rahtz*, 1999}
{see *Knuth*, 1986, chap. 2}

```
\usepackage[curly,comma]{natbib}
\bibliographystyle{agu}
\citet{LGC97} \\ \citep{Knuth-CT-a,LWC99} \\
\citep[see][chap.~2]{Knuth-CT-a}
```

12-3-21

Yet another method to customize the layout is mainly intended for package and/or class file writers: the `\bibpunct` declaration. It takes seven arguments (the first optional) that define various aspects of the citation format. It is typically used to define the default citation format for a particular BIBTEX style. For example, the natbib package contains many definitions like this:

```
\newcommand\bibstyle@chicago{\bibpunct{(}{)}{;}{a}{,}{,}}
```

That definition will be selected when you choose chicago as your BIBTEX style or when you specify it as the argument to `\citestyle`. Similar declarations can be added for BIBTEX styles that natbib does not directly support. This effect is most readily realized by grouping such declarations in the local configuration file `natbib.cfg`. For details on the meanings of the arguments, see the documentation accompanying the natbib package.

If there are conflicting specifications, then the following rules apply: the lowest priority is given to internal `\bibstyle@`⟨*name*⟩ declarations, followed by the options specified in the `\usepackage` declarations. Both are overwritten by an explicit `\bibpunct` or `\citestyle` declaration in the document preamble.

Forcing all author names on a single line

Normally, natbib does not prevent a line break within the author list of a citation. By specifying the option nonamebreak, you can ensure that all author names in one citation will be kept on a single line. In normal circumstances this is seldom a good idea as it is likely to cause overfull hboxes, but it helps with some hyperref problems.

[1]Despite its name this option will produce a ";" semicolon.

Customizing the bibliography layout

The thebibliography environment, as implemented by natbib, automatically adds a heading before the list of publications. By default, natbib selects an unnumbered heading of the highest level, such as \chapter* for a book type class or \section* for the article class or a variant thereof. The actual heading inserted is stored in the command \bibsection. Thus, to modify the default, you have to change its definition. For instance, you can suppress the heading altogether or choose a numbered heading.

For one particular situation natbib offers direct support: if you specify the option sectionbib, you instruct the package to use \section*, even if the highest sectional unit is \chapter. This option is useful if natbib and chapterbib are used together (see Section 12.6.1).

Between \bibsection and the start of the list, natbib executes the hook \bibpreamble, if defined. It allows you to place some text between the heading and the start of the actual reference list. It is also possible to influence the font used for the bibliography by defining the command \bibfont. This hook can also be used to influence the list in other ways, such as setting it unjustified by adding \raggedright. Note that both \bibpreamble and \bibfont are undefined by default (and thus need \newcommand), while \bibsection needs redefining with \renewcommand.

Finally, two length parameters are available for customization. The first line in each reference is set flush left, and all following lines are indented by the value stored in \bibhang (default 1em). The vertical space between the references is stored in the rubber length \bibsep (the default value is usually equal to \itemsep as defined in other lists).

To show the various possibilities available we repeat Example 12-1-2 on page 685 but apply all kinds of customization features (not necessarily for the better!). Note the presence of \par at the end of \bibpreamble. Without it the settings in \bibfont would affect the inserted text!

Entries with multiple authors might be problematical, e.g., Goossens et al. [1997a] and Goossens et al. [1997b] or even Goossens et al. [1997a,b]. But then they might not.

1 References

Some material inserted between heading and list.

M. Goossens, S. Rahtz, and F. Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997a. ISBN 0-201-85469-4.

M. Goossens, B. User, J. Doe, et al. Ambiguous citations. Submitted to the IBM J. Res. Dev., 1997b.

12-3-22

```
\usepackage{natbib}
\bibliographystyle{abbrvnat}
\renewcommand\bibsection{\section{\refname}}
\newcommand\bibpreamble{Some material
     inserted between heading and list.\par}
\newcommand\bibfont
                 {\footnotesize\raggedright}
\setlength\bibhang{30pt}
\setlength\bibsep{1pt plus 1pt}
Entries with multiple authors might be
problematical, e.g., \cite{LGC97} and
\cite{test97} or even \cite{LGC97,test97}.
But then they might not.
\bibliography{tex}
```

Publications without author or year information

To use the author-date citation system, the entries in your list of publications need to contain the necessary information. If some information is missing, citations with `\citet` or its variants may produce strange results.

If the publication has no author but an editor, then most BibTeX styles will use the latter. However, if both are missing, the solutions implemented differ greatly. BibTeX files in "Harvard" style (e.g., agsm) use the first three letters from the key field if present; otherwise, they use the first three letters from the organization field (omitting "The␣" if necessary); otherwise, they use the full title. If an entry has no year, then "n.d." is used. This will result in usable entries except in the case where part of the key field is selected:

Koppitz (n.d.) / *TUGboat The Communications of the TEX User Group* (1980ff) / mak (2000)

```
\usepackage{natbib} \bibliographystyle{agsm}
\citet{G-G} / \citet{oddity} / \citet{GNUMake}
```

12-3-

With the same entries, BibTeX styles produced with makebst (e.g., unsrtnat) use the following strategy: if a key field is present, the whole field is used as an "author"; otherwise, if an organization field is specified, its first three letters are used (omitting "The␣" if necessary); otherwise, the first three letters of the citation label are used. A missing year is completely omitted. In case of textual citations, this means that only the author name is printed. In that situation, or when the key field is used, it is probably best to avoid `\citet` and always use `\citep` to make it clear to the reader that you are actually referring to a publication and not just mentioning some person in passing.

Koppitz / odd [1980ff] / make
[Koppitz] / [odd, 1980ff] / [make]

```
\usepackage{natbib} \bibliographystyle{unsrtnat}
\citet{G-G} / \citet{oddity} / \citet{GNUMake} \\
\citep{G-G} / \citep{oddity} / \citep{GNUMake}
```

12-3-2

As a final example we show the results when using the chicago BibTeX style. Here the GNU manual comes out fine (the full organization name is used), but the entry with the date missing looks odd.

Koppitz (Koppitz) / odd (80ff) / Free Software Foundation (2000)
(Koppitz, Koppitz) / (odd, 80ff) / (Free Software Foundation, 2000)

```
\usepackage{natbib} \bibliographystyle{chicago}
\citet{G-G} / \citet{oddity} / \citet{GNUMake} \\
\citep{G-G} / \citep{oddity} / \citep{GNUMake}
```

12-3-2

Forcing author-date style

The natbib package produces author-date citations by default, when used together with most BibTeX styles. You can also explicitly request the author-date system by loading the package with the option authoryear.

However, for this approach to work, it is important that the BibTeX style passes author-date information back to the document. Hence, .bst files, such as LaTeX's plain, which have been developed for numerical citation systems only, are unable to transfer this information. In that case natbib will ignore the authoryear option and, if you use \citet or one of its variants, you get warnings about missing author information and output similar to the following:

12-3-26 **(author?)** [3] / **(author?)** [1] / **(author?)** [2]

```
\usepackage{natbib} \bibliographystyle{plain}
\citet{G-G} / \citet{oddity} / \citet{GNUMake}
```

Here it is best to switch to a BIBTEX style that supports the author-date system, such as plainnat instead of plain.

Indexing citations automatically

Citations can be entered in the index by inserting a \citeindextrue command at any point in the document. From that point onward, and until the next \citeindexfalse (or the end of the current group) is encountered, all variants of the \citet and \citep commands will generate entries in the index file (if one is written). With \citeindextrue in effect, the \bibitem commands in the thebibliography environment will also generate index entries. If this result is not desired, issue a \citeindexfalse command before entering the environment (e.g., before calling \bibliography).

The index format is controlled by the internal command \NAT@idxtxt. It has the following default definition:

```
\newcommand\NAT@idxtxt{\NAT@name\ \NAT@open\NAT@date\NAT@close}
```

Thus, it produces entries like "Knuth (1986)". For citations without author or year information the results will most likely come out strangely. The citations in Example 12-3-24 will generate the following entries:

```
\indexentry{{Koppitz}\ []}{6}
\indexentry{{odd}\ [1980ff]}{6}
\indexentry{{make}\ []}{6}
```

If you want to redefine the command, for example, to just generate the author's name, you can do so in the file natbib.cfg or in the preamble of your document. In the latter case, do not forget \makeatletter and \makeatother!

It is also possible to produce a separate index of citations by using David Jones's index package (see Section 11.4.3). It allows you to generate multiple index lists using the \newindex command. For this to work you must first declare the list and then associate automatic citation indexing with this list in the preamble:

```
\usepackage{index}
\newindex{default}{idx}{ind}{Index}              % the main index
\newindex{cite}{cdx}{cnd}{Index of Citations}
\renewcommand\citeindextype{cite}
```

Later on use \printindex[cite] to indicate where the citation index should appear in the document.

BIBTEX styles for natbib

As mentioned in the introduction, natbib was developed to work with various BIBTEX styles that implement some form of author-date scheme. In addition to those third-party styles, natbib works with all styles that can be produced with the custom-bib bundle (see Section 13.5.2 on page 798). It is distributed with three styles—abbrvnat, plainnat, and unsrtnat—that are extensions of the corresponding standard styles. They have been adapted to work better with natbib, allowing you to use some of its features that would be otherwise unavailable. These styles also implement a number of extra fields useful in the days of electronic publications:

doi For use with electronic journals and related material. The Digital Object Identifier (DOI) is a system for identifying and exchanging intellectual property in the digital environment, and is supposedly more robust than URLs (see http://www.doi.org for details). The field is optional.

eid As electronic journals usually have no page numbers, they use a sequence identifier (EID) to locate the article within the journal. The field is optional and will be used in place of the page number if present.

isbn The International Standard Book Number (ISBN), a 10-digit unique identification number (see www.isbn.org). The ISBN is defined in ISO Standard 2108 and has been in use for more than 30 years. The field is optional.

issn The International Standard Serial Number (ISSN), an 8-digit number that identifies periodical publications (see www.issn.org). The field is optional.

url The Uniform Resource Locator (URL) for identifying resources on the web. The field is optional. As URL addresses are typically quite long and are set in a typewriter font, line-breaking problems may occur. They are therefore automatically surrounded with a \url command, which is given a simple default definition if undefined. Thus, by using the url package (see Section 3.1.8), you can drastically improve the line-breaking situation as then URLs can be broken at punctuation marks.

12.3.3 bibentry—Full bibliographic entries in running text

Instead of grouping all cited publications in a bibliography, it is sometimes required to directly typeset the full information the first time a publication is referenced. To help with this task Patrick Daly developed the bibentry package as a companion to the natbib package.

`\nobibliography{`*BibTeX-database-list*`}` `\bibentry{`*key*`}`

This command works as follows: instead of the usual `\bibliography` command, which loads the `.bbl` file written by BibTeX and typesets the bibliography, you use `\nobibliography` with the same list of BibTeX database files. This command will read the `.bbl` and process the information, so that references to entries can be made elsewhere in the document. To typeset a citation with the full bibliographical information, use `\bibentry`. The usual author-date citation can be produced with any of the natbib commands. Here is an example:

For details see Knuth, D. E., Typesetting Concrete Mathematics, *TUGboat, 10*, 31–36, 1989. General information can be found in Knuth, D. E., *The TeXbook*, vol. A of *Computers and Typesetting*, Addison-Wesley, Reading, MA, USA, 1986.

12-3-27 As shown by *Knuth* [1989] ...

```
\usepackage{bibentry,natbib}
\bibliographystyle{agu}

\raggedright \setlength\parindent{12pt}
\nobibliography{tex}
For details see \bibentry{Knuth:TB10-1-31}.
General information can be found in
\bibentry{Knuth-CT-a}.

As shown by \citet{Knuth:TB10-1-31} \ldots
```

There are a number of points to be noted here: the `\nobibliography` command must be placed inside the body of the document but before the first use of a `\bibentry` command. In the preamble a `\nobibliography` will be silently ignored, and any `\bibentry` command used before it will produce no output. Such a command is therefore best placed directly after `\begin{document}`. *Potential pitfalls*

Another potential problem relates to the choice of BibTeX style. The bibentry package requires the entries in the `.bbl` file to be of a certain form: they must be separated by a blank line, and the `\bibitem` command must be separated from the actual entry text by either a space or a newline character. This format is automatically enforced for BibTeX styles produced with makebst but other BibTeX styles may fail, including some that work with natbib.

The `\bibentry` command automatically removes a final period in the entry so that the reference can be used in mid-sentence. However, if the entry contains other punctuation, such as a period as part of a note field, the resulting text might still read strangely. In that case the only remedy might be to use an adjusted BibTeX database entry.

One can simultaneously have a bibliography and use the `\bibentry` command to produce full citations in the text. In that case, place the `\bibliography` command to produce the bibliography list at the point where it should appear. Directly following `\begin{document}`, add the command `\nobibliography*`. This variant takes no argument, because the BibTeX database files are already specified on the `\bibliography` command. As a consequence, all publications cited with `\bibentry` will also automatically appear in the bibliography, because a single `.bbl` file is used.

12.4 The author-number system

As mentioned in the introduction, currently there exists no BIBTEX style file that implements the author-number system for documents in which the publications should be numbered individually for each author. If, however, the publications are numbered sequentially throughout the whole bibliography, then ample support is provided by BIBTEX and by the natbib package already encountered in conjunction with the author-date system.

12.4.1 natbib—Revisited

Although originally designed to support the author-date system, natbib is also capable of producing author-number and number-only references. Both types of references are provided with the help of BIBTEX styles specially designed for numbered bibliographies, similar to the BIBTEX styles normally used for the author-date style of citations.

By default, natbib produces author-date citations. If you are primarily interested in citing references according to the number-only or author-number system, load natbib with the `numbers` option.

For comparison, we repeat Example 12-3-5 on page 701 with the `numbers` option loaded. This option automatically implies the options `square` and `comma`; thus, if you prefer round parentheses, use the option `round` and overwrite the default choice.

Goossens et al. [1]
Goossens et al. [1, chap. 2]
Goossens et al. [see 1, chap. 2]
pre-note only: Goossens et al. [see 1]

[1]
[1, chap. 2]
[see 1, chap. 2]
pre-note only: [see 1]

```
\usepackage[numbers]{natbib}
\citet{LGC97}                        \\
\citet[chap.~2]{LGC97}               \\
\citet[see][chap.~2]{LGC97}          \\
pre-note only: \citet[see][]{LGC97}  \\[5pt]
\citep{LGC97}                        \\
\citep[chap.~2]{LGC97}               \\
\citep[see][chap.~2]{LGC97}          \\
pre-note only: \citep[see][]{LGC97}
```

12-4-1

As you can see, the `\citet` command now generates citations according to the author-number system, while `\citep` produces number-only citations. In fact, if natbib is set up to produce numerical citations, LATEX's `\cite` command behaves like `\citep`. In author-date mode, natbib makes this command act as short form for the command `\citet`.

All variant forms of `\citet` and `\citep`, as discussed in Section 12.3.2, are also available in numerical mode, though only a few make sense. For example, `\citep*` gives the same output as `\citep`, because there are no authors inside the parentheses.

Goossens, Rahtz, and Mittelbach [1]
Goossens et al.
Goossens, Rahtz, and Mittelbach
1997 or [1997]

```
\usepackage[numbers]{natbib}
\citet*{LGC97}                 \\
\citeauthor{LGC97}             \\
\citeauthor*{LGC97}            \\
\citeyear{LGC97}  or  \citeyearpar{LGC97}
```

12-4-2

The commands `\citealt` and `\citealt*` should probably not be used, as without the parentheses the citation number is likely to be misinterpreted. However, in certain situations `\citealp` might be useful to obtain that number on its own and then perhaps use it together with `\citetext`.

Goossens et al. 1
Goossens, Rahtz, and Mittelbach 1
1
1, p. 236 etc.

```
\usepackage[numbers]{natbib}
\citealt{LGC97}                  \\
\citealt*{LGC97}                 \\
\citealp{LGC97}                  \\
\citealp[p.~236]{LGC97} etc.
```

12-4-3

Some journals use numerical citations with the numbers raised as superscripts. If loaded with the option super, the natbib package supports this type of citation. In that case our standard example (compare with Example 12-4-1) will produce the following:

Goossens et al.[1]
Goossens et al.[1], chap. 2
Goossens et al. see[1], chap. 2
pre-note only: Goossens et al. see[1]
[1]
[1] (chap. 2)
[1] (chap. 2)
pre-note only:[1]

```
\usepackage[super]{natbib}
\citet{LGC97}                              \\
\citet[chap.~2]{LGC97}                     \\
\citet[see][chap.~2]{LGC97}                \\
pre-note only: \citet[see][]{LGC97}        \\[5pt]
\citep{LGC97}                              \\
\citep[chap.~2]{LGC97}                     \\
\citep[see][chap.~2]{LGC97}                \\
pre-note only: \citep[see][]{LGC97}
```

12-4-4

As you will observe, the use of the optional arguments produces somewhat questionable results; in the case of `\citep` the *pre-note* will not appear at all. Thus, with this style of citation, it is usually best to stick to the basic forms of any such commands.

For superscript citations natbib removes possible spaces in front of the citation commands so as to attach the number to the preceding word. However, in contrast to the results produced with the cite package, punctuation characters will not migrate in front of the citation, nor is there any check for double periods. To illustrate this we repeat Example 12-2-11 from page 696.

...Knuth's book[2]; see also Goossens et al.[1].
...Knuth's book;[2] see also Goossens et al.[1]

```
\usepackage[super]{natbib}
\ldots Knuth's book~\citep{Knuth-CT-a}; see also \citet{LGC97}.
\par %%% Manually corrected in two places:
\ldots Knuth's book;\citep{Knuth-CT-a}  see also \citet{LGC97}
```

12-4-5

The packages natbib and cite are unfortunately incompatible (both modify LaTeX's internal citation mechanism), so in cases like Example 12-4-5 you have to change the input if natbib is to be used.

Sorting and compressing numerical citations

As seen in Section 12.2.2 the cite package sorts multiple citations and optionally compresses them into ranges. This feature is also implemented by natbib and can be activated through the options sort and sort&compress.

We have already encountered sort in connection with author-date citations. With numerical citations (i.e., the options numbers and super), the numbers are sorted. To show the effect we repeat Example 12-2-5 from page 693, except that we omit the undefined citation.

Good information about TeX and LaTeX can be found in [1, 2, 3, 4].

```
\usepackage[sort]{natbib} \bibliographystyle{plain}
Good information about \TeX{} and \LaTeX{} can be found in
\citep{LGC97,LWC99,Knuth-CT-a,Knuth:TB10-1-31}.
```

12-4-6

With the option sort&compress, the numbers are not only sorted but also compressed into ranges if possible. In author-date citation mode, this option has the same effect as sort.

Good information about TeX and LaTeX can be found in [1–4].

```
\usepackage[sort&compress]{natbib}\bibliographystyle{plain}
Good information about \TeX{} and \LaTeX{} can be found in
\citep{LGC97,LWC99,Knuth-CT-a,Knuth:TB10-1-31}.
```

12-4-7

The rules for selecting numerical mode

As mentioned previously, natbib, by default, works in author-date mode. However, for the previous two examples, natbib selected numerical mode without being explicitly told to do so (via the numbers or super option). This result occurs because the plain BibTeX style does not carry author-date information in the \bibitem commands it generates. Whenever there is a single \bibitem without the relevant information, natbib automatically switches to numerical mode. Even specifying the option authoryear will not work in that case.

If a BibTeX style supports author-date mode, then switching to numerical mode can be achieved by one of the following methods, which are listed here in increasing order of priority:

1. By selecting a \bibliographystyle with a predefined numerical citation style (e.g., defined in a local configuration file, or in a class or package file).
2. By specifying the option numbers or super, as shown in most examples in this section.
3. By explicitly using \bibpunct with the fourth mandatory argument set to n or s (for details, see the package documentation).

4. By explicitly using `\citestyle` with the name of a predefined numerical bibliography style.

Customizing natbib in numerical mode

The majority of options and parameters to customize natbib have already been discussed on pages 705–707, but in numerical mode there are two more commands available to modify the produced layout. By default, citation numbers are typeset in the main body font. However, if you define `\citenumfont` (as a command with one argument), it will format the citation number according to its specification.

Similarly, you can manipulate the format of the number as typeset within the bibliography by redefining `\bibnumfmt` using `\renewcommand`.[1] The default definition for this command usually produces square brackets around the number.

Images are discussed elsewhere, see (**1**, **2**).

References

1. M. Goossens, S. Rahtz, and F. Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997. ISBN 0-201-85469-4.

2. D. E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0.

12-4-8

```
\usepackage[numbers,round]{natbib}
\bibliographystyle{abbrvnat}
\newcommand\bibfont{\small\raggedright}
\setlength\bibhang{30pt}          % ignored!
\setlength\bibsep{1pt plus 1pt}
\newcommand\citenumfont[1]{\textbf{#1}}
\renewcommand\bibnumfmt[1]{\textbf{#1.}}
Images are discussed elsewhere,
see \citep{LGC97,Knuth-CT-a}.

\bibliography{tex}
```

While `\bibsection`, `\bibpreamble`, `\bibfont`, and `\bibsep` work as before, the parameter `\bibhang` has no effect, since in a numbered bibliography the indentation is defined by the width of the largest number.

12.5 The short-title system

12.5.1 jurabib—Customizable short-title references

Classifying the jurabib package developed by Jens Berger as a package implementing the short-title system is not really doing it justice (no pun intended), as in fact it actually supports other citation systems as well.

Besides short-title citations it offers support for author-date citations (by providing the natbib command interface), various options to handle specific requirements from the humanities, and special support for citing juridical works such as commentaries (hence the name jurabib).

[1] The package is unfortunately somewhat inconsistent in providing or not providing defaults for the customization hooks. This means that you have to use either `\newcommand` or `\renewcommand` depending on the context.

The package uses an extended option concept where options are specified with a "*key=value*" syntax. The package supports more than 30 options, each of which may be set to a number of values, covering various aspects of presenting the citation layout in the text and the references in the bibliography. In this book we can show only a small selection of these possibilities. For further information refer to the package documentation, which is available in English and German.

Default used for all examples in this section!

It is inconvenient to handle so many options as part of the `\usepackage` declaration, so jurabib offers the `\jurabibsetup` command as an alternative. It can be used in the preamble or in the package configuration file `jurabib.cfg` (to set the defaults for all documents). Settings established when loading the package or via `\jurabibsetup` in the preamble will overwrite such global defaults. For the examples in this section we will use the following defaults

```
\jurabibsetup{titleformat=colonsep,commabeforerest=true}
```

and extend or overwrite them as necessary. Their meaning is explained below.

In contrast to natbib, the jurabib package requires the use of specially designed BIBTEX style files. It expects a `\bibitem` command with a specially structured optional argument to pass all kinds of information back to the user-level citation commands (see page 699). These BIBTEX styles also implement a number of additional fields useful in conjunction with jurabib.

To show the particular features of jurabib, we use the small BIBTEX database shown in Figure 12.3 on the facing page together with the database used previously (Figure 12.2 on page 690). If not explicitly documented otherwise, all examples in this section have the line

```
\newpage\bibliography{tex,jura}
```

implicitly appended at the end when processed.

The basic syntax

Like the natbib package, the jurabib package extends the standard LATEX citation command `\cite` with a second optional argument.

`\cite`[*post-note*]{*key(s)*} `\cite`[*annotator*][*post-note*]{*key(s)*}

If two optional arguments are present, then the *post-note* argument moves to the second position, the same behavior found with the natbib syntax. But in the default set-up there is a big difference in that we do not have a *pre-note* argument but rather an *annotator* argument provided for a citation method used in legal works.[1] In that discipline, works often have an original author (under which the work is listed in the bibliography) as well as annotators who provide commentaries in the particular edition. These annotators are mentioned in the citation

[1]See page 721 if you want it to be a *pre-note* instead.

```
@BOOK{zpo,
  author     = {Adolf Baumbach and Wolfgang Lauterbach
                and Jan Albers and Peter Hartmann},
  title      = {Zivilproze\ss ordnung mit
                Gerichtsverfassungsgesetz und anderen
                Nebengesetzen},
  shorttitle = {ZPO},
  language   = {ngerman},
  edition    = {59. neubearb.},
  year       = 2002,
  address    = {M\"unchen}
}
@BOOK{aschur,
  author     = {Hans Brox and Wolf-Dietrich Walker},
  title      = {Allgemeines Schuldrecht},
  language   = {ngerman},
  edition    = {29.},
  year       = 2003,
  address    = {M\"unchen}
}
@BOOK{bschur,
  author     = {Hans Brox and Wolf-Dietrich Walker},
  title      = {Besonderes Schuldrecht},
  shorttitle = {BSchuR},
  language   = {ngerman},
  edition    = {27.},
  year       = 2002,
  address    = {M\"unchen}
}
@BOOK{bgb,
  author     = {Otto Palandt},
  shortauthor= {Otto Palandt},
  title      = {B\"urgerliches Gesetzbuch},
  shorttitle = {BGB},
  language   = {ngerman},
  edition    = {62.},
  year       = 2003,
  publisher  = {Beck Juristischer Verlag},
  address    = {M\"unchen}
}
```

Figure 12.3: Sample BibTeX database `jura.bib`

but not in the bibliography. Without further adjustments a citation will list only the author surnames (separated by slashes if there are several authors), followed by the *annotator* if present, followed by a possible *post-note.* If the BibTeX entry contains a `shortauthor` field, then it is used instead of the surnames. If you want to specify an *annotator*, use an empty *post-note.* By default, a title or short title is shown only if the author is cited with different works in the same document.

Brox/Walker
Brox/Walker, § 123
Otto Palandt/Heinrichs
Otto Palandt/Heinrichs, § 26

12-5-1

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\cite{aschur}                  \\
\cite[\S\,123]{aschur}         \\
\cite[Heinrichs][]{bgb}        \\
\cite[Heinrichs][\S\,26]{bgb}
```

As you see, there is no way to determine from the typeset result that "Walker" is a co-author but "Heinrichs" is an annotator. To make this distinction immediately visible, jurabib offers a number of options implementing common citation styles. You can, for example, change the font used for the annotator, or change the separator between author and annotator. Both of these changes have been specified in the first part of the next example. You can also move the annotator before the author, a solution shown in two variants in the second part of the example.

Brox/Walker
Otto Palandt–*Heinrichs*, § 26
Heinrichs, Otto Palandt, § 26
Heinrichs in: Otto Palandt, § 26

12-5-2

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{annotatorformat=italic,annotatorlastsep=divis}
\cite{aschur}  \\ \cite[Heinrichs][\S\,26]{bgb} \\
\jurabibsetup{annotatorfirstsep=comma}
\cite[Heinrichs][\S\,26]{bgb} \\
\jurabibsetup{annotatorfirstsep=in,annotatorformat=normal}
\cite[Heinrichs][\S\,26]{bgb}
```

Another way to clearly distinguish authors and annotators is to use the option authorformat with the keyword and (which replaces slashes with commas and "and"), the keyword dynamic (in which case different fonts are used depending on whether an *annotator* is present), or the keyword year (which moves the publication year directly after the author). The authorformat option can also be used to influence other aspects of the formatting of author names. Some examples are shown below. A complete list of allowed keywords is given in the package documentation. Note that if you use several keywords together (as done below), you need an additional set of braces to indicate to jurabib where the keyword list ends and the next option starts.

BROX and WALKER
OTTO PALANDT/HEINRICHS, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat={and,smallcaps}}
\cite{aschur}  \\ \cite[Heinrichs][\S\,26]{bgb} \par
```

12-5-3

If the keyword dynamic is used, the annotator's name is set in italics while the original author's name is set in the body font.[1] For works without an annotator, author names are set in italics. One can think of this style as labeling those people who have actually worked on the particular edition.

Brox/Walker
Otto Palandt/*Heinrichs*, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat=dynamic}
\cite{aschur}  \\ \cite[Heinrichs][\S\,26]{bgb} \par
```

12-5-4

The keywords and, dynamic, and year can be combined, while smallcaps and italic contradict each other with the last specification winning:

Brox and *Walker* (2003)
Otto Palandt (2003)/*Heinrichs*, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat={and,smallcaps,year,italic}}
\cite{aschur}  \\ \cite[Heinrichs][\S\,26]{bgb} \par
```

12-5-5

The information passed back by BIBTEX is very detailed and structured into individual fields whose contents can be accessed using the \citefield command.

\citefield[*post-note*]{*field*}{*key(s)*}

The *field* argument is one of the following fields from the BIBTEX database entry referenced by the *key* argument: author, shortauthor, title, shorttitle, url, or year. It can also be apy (address-publisher-year combination).

[1]The fonts used can be customized by redefining the commands \jbactualauthorfont and \jbactualauthorfontifannotator.

Whether more than a single *key* is useful is questionable for most fields. Indeed, even with `\cite` multiple keys are seldom useful unless no optional arguments are present.

BROX, HANS/WALKER, WOLF-DIETRICH
BSchuR, § 53
Reading, MA, USA: Addison-Wesley Longman, 1997
Allgemeines Schuldrecht; Besonderes Schul-
12-5-6 drecht

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat=smallcaps}
\citefield{author}{aschur}                   \\
\citefield[\S\,53]{shorttitle}{bschur} \\
\citefield{apy}{LGC97}                       \\
\citefield{title}{aschur,bschur}
```

If you are familiar with the German language, you will notice that the hyphenation of "Schul-drecht" is incorrect: it should have been "Schuld-recht". How to achieve this hyphenation automatically is explained on page 733.

Citations with short and full titles

As mentioned before, by default jurabib does not include a title in the citation text. The exception occurs when there are several works cited by the same author, so that a title is necessary to distinguish between them. This behavior can be changed in several ways, but first we have a look at the "title" that will be used:

Brox/Walker: Allgemeines Schuldrecht
Brox/Walker: BSchuR
Knuth: The TeXbook
12-5-7 Knuth: TUGboat 10 [1989]

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\cite{aschur}      \\  \cite{bschur}      \\[2pt]
\cite{Knuth-CT-a} \\  \cite{Knuth:TB10-1-31}
```

If you compare the first two lines of the previous example with the BIBTeX database files listed in Figure 12.3 on page 717, you see that the `shorttitle` field was used if available; otherwise, the `title` field was used. In fact, you will get a warning from jurabib for this adjustment: "shorttitle for aschur is missing - replacing with title". A different approach is taken for entries of type `article` or `periodical`; there, a missing `shorttitle` is replaced by the journal name, volume number, and year of publication, which is why we got "TUGboat 10 [1989]".

`\citetitle`[*post-note*]{*key(s)*} `\citetitle`[*annotator*] [*post-note*]{*key(s)*}
`\cite*`[*post-note*]{*key(s)*} `\cite*`[*annotator*] [*post-note*]{*key(s)*}

To force the production of a title in the citation, you can use `\citetitle` instead of `\cite`. To leave out the title, you can use `\cite*`. You should, however, be aware that the latter command can easily lead to ambiguous citations, as shown in the next example.

Baumbach et al.: ZPO, Brox/Walker, and Brox/
12-5-8 Walker are three different books, or not?

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\citetitle{zpo}, \cite*{aschur}, and \cite*{bschur}
are three different books, or not?
```

Also note that this meaning of `\cite*` is quite different from its use in natbib (where it denotes using a full list of authors). If you switch between both packages depending on the circumstances, it might be better to avoid it altogether.

`\citetitleonly[`*post-note*`]{`*key*`}`

It is also possible to refer to only the title, including a *post-note* if desired.

ZPO, § 13

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\citetitleonly[\S\,13]{zpo}
```

12-5-9

Getting short-title citations automatically

Short-title citations can be generated by default by specifying the option `titleformat` and the keyword `all`. Like `authorformat`, this option can take several keywords. We already know about `colonsep`, which we used as a default setting for all the examples. In the next example we overwrite it with `commasep` and print the titles in `italic`.

Brox/Walker, *Allgemeines Schuldrecht*, § 123
Brox/Walker, *BSchuR*
Otto Palandt/Heinrichs, *BGB*
Knuth, *TUGboat 10 [1989]*

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{titleformat={all,commasep,italic}}
\cite[\S\,123]{aschur}  \\ \cite{bschur}   \\
\cite[Heinrichs][]{bgb} \\ \cite{Knuth:TB10-1-31}
```

12-5-10

`\citetitlefortype{`*BIBTEX-type-list*`}` `\citenotitlefortype{`*BIBTEX-type-list*`}`

Instead of citing all works with titles you can select short-title citations based on a particular BIBTEX type. For example,

```
\citetitlefortype{article,book,manual}
```

would reference these three types with the title and all other publication types without it, unless the author is cited with several works. Since such a list can grow quite large, alternatively you can select automatic title citations for all works (with `titleformat`) and then specify those types that should have no titles when referenced. This is done in the next example for the type `book`. Nevertheless, the book by Knuth is cited with its title, since we also cite an article by him.

Brox/Walker
Goossens/Rahtz
Knuth: The TEXbook
Knuth: TUGboat 10 [1989]

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{titleformat=all} \citenotitlefortype{book}
\cite{bschur}      \\  \cite{LWC99}           \\
\cite{Knuth-CT-a} \\  \cite{Knuth:TB10-1-31}
```

12-5-11

Indexing citations automatically

The author names in citations can be entered in the index by using the option `authorformat` with the keyword `indexed`. By default, this is done only for citations inside the text; authors referred to only in the bibliography are not listed. This behavior can be changed by setting `\jbindexbib` in the preamble or in a configuration file. For formatting the index entries, `\jbauthorindexfont` is available. For example,

```
\renewcommand\jbauthorindexfont[1]{\textit{#1}}
```

means that the author names will appear in italic in the index.

Instead of placing the author names in the main index, you can produce a separate author index by loading the index package (see Section 11.4.3) and then using a construction like

```
\usepackage{index}
\newindex{default}{idx}{ind}{Index}            % the main index
\newindex{authors}{adx}{and}{Index of Authors}
\renewcommand\jbindextype{authors}
```

in the preamble, and later on `\printindex[authors]` to indicate where the author index should appear in the document.

No support is available for more elaborate indexes as required for some types of law books (e.g., "Table of Cases" or "Table of Statutes"). If this is required, consider using the camel package instead of jurabib.

Using natbib citation semantics

The optional *annotator* argument is useful only in legal studies. In other disciplines, it is more common to require a *pre-note* (e.g., "compare..."). To account for this, the meanings of the optional arguments can be modified by loading the package with the option `see`.

`\cite[`*pre-note*`][`*post-note*`]{`*key(s)*`}` (with option `see`)

The `see` option replaces the default *annotator* optional argument with a *pre-note* argument in case two optional arguments are used. The `\cite` command then has the same syntax and semantics as it does with the natbib package.

(Goossens/Rahtz/Mittelbach)
(Goossens/Rahtz/Mittelbach, chap. 2)
(compare Goossens/Rahtz/Mittelbach)
12-5-12 (see Goossens/Rahtz/Mittelbach, chap. 2)

```
\usepackage[see,round]{jurabib}
\bibliographystyle{jurabib}
\cite{LGC97}                  \\
\cite[chap.~2]{LGC97}         \\[3pt]
\cite[compare][]{LGC97}       \\
\cite[see][chap.~2]{LGC97}
```

This work was cited as ...

When using a short-title system for citations (e.g., by setting `titleformat` to `all`), it can be helpful to present the reader with a mapping between the full entry and the short title. This is commonly done by displaying the short title in parentheses at the end of the corresponding entry in the bibliography. The jurabib package supports this convention with the option `howcited`. It can take a number of keywords that configure the mechanism in slightly different ways. For example, the keyword `all` instructs the package to add "how cited" information to all entries in the bibliography. Thus, if we add to Example 12-5-10 on page 720 the line

```
\jurabibsetup{howcited=all}
```

we will get the following bibliography listing. Note that the short title is formatted in exactly the same way as it will appear in the citation.

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27th edition. München, 2002 (cited: Brox/Walker, *BSchuR*)

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29th edition. München, 2003 (cited: Brox/Walker, *Allgemeines Schuldrecht*)

Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207 (cited: Knuth, *TUGboat 10 [1989]*)

Palandt, Otto: Bürgerliches Gesetzbuch. 62th edition. München: Beck Juristischer Verlag, 2003 (cited: Otto Palandt, *BGB*) 12-5-13

However, it is usually not necessary to display for all entries how they are cited. For articles, the short-title citation is always "author name, journal, volume, and year". If a work is cited with its full title (i.e., if there is no `shorttitle` field) or if only a single publication is cited for a certain author, then the reader will generally be able to identify the corresponding entry without any further help. To allow for such a restricted type of "back-references", jurabib offers the keywords `compare`, `multiple`, and `normal`.

If you use `compare`, then a back-reference is created only if the entry contains a `shorttitle` field and the `title` and `shorttitle` fields differ. With respect to Example 12-5-13 this means that only the first and last entries would show the back-references.

If you use `multiple` instead, then back-references are generated whenever an author is cited with several works except for citations of articles. In the above example, the first two entries would get back-references. If we also had a citation to `Knuth-CT-a`, then it would also show a back-reference, while Knuth's article in *TUGboat* would be still without one.

Both keywords can be used together. In that case back-references are added to entries for authors with several publications as well as to entries whose short titles differ from their main titles.

Finally, there is the keyword `normal` (it is also used if you specify the option without a value). This keyword works slightly differently from the others in that it needs support to be present in the BibTeX database. If it is used, an entry gets a back-reference if and only if the BibTeX field howcited is present. The field can have two kinds of values. If it has a value of "1", the back-reference lists exactly what is shown in the citation in text. With any other value, the actual contents of the howcited field are used for the back-reference, including any formatting directives contained therein.

The text surrounding the back-reference can be customized by redefining the commands \howcitedprefix and \howcitedsuffix. In addition, you can specify what should happen with entries that have been added via \nocite by changing \bibnotcited (empty by default). Because these commands may contain text that should differ depending on the main language of the document, they are redefined using a special mechanism (\AddTo) that is explained on page 733.

... Brox/Walker: BSchuR ... Knuth ...

References

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27th edition. München, 2002 (cited as Brox/Walker: BSchuR).

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29th edition. München, 2003 (not cited).

Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207 (cited as Knuth).

12-5-14

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{howcited=all}
\AddTo\bibsall{%
  \renewcommand\howcitedprefix
                        { (cited as }%
  \renewcommand\howcitedsuffix{).}%
  \renewcommand\bibnotcited
                        { (not cited).}}
\nocite{aschur}
\ldots \cite{bschur} \ldots
\cite{Knuth:TB10-1-31} \ldots
\bibliography{jura,tex}
```

Full citations inside the text

While producing full citations inside the text with natbib requires a separate package and some initial preparation, this citation method is fully integrated in jurabib. The complete entry can be shown for one or more individual citations, for all citations, or automatically for only the first citation of a work. This citation method is most often used in footnotes; see page 726 for information on how to automatically arrange footnote citations.

\fullcite[*post-note*]{*key(s)*}   \fullcite[*annotator*][*post-note*]{*key(s)*}

This command works like \cite but displays the full bibliographical data. The *annotator*, if present, will be placed in front of the citation just as if annotatorfirstsep=in had been specified.

Compare the next example with Example 12-3-27 from page 711. The keyword citationreversed arranges for the author name to appear with surname last (in the bibliography the surname comes first). Related keywords are allreversed (surname last in text and bibliography) and `firstnotreversed` (surname first for first author, last for all others in multiple-author works).

For details see Donald E. Knuth: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896–3207. General information can be found in Donald E. Knuth: The TEXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0.

As shown by Knuth (1989) ...

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{authorformat=citationreversed}

\raggedright \setlength\parindent{12pt}
For details see \fullcite{Knuth:TB10-1-31}.
General information can be found in
\fullcite{Knuth-CT-a}.

As shown by \citet{Knuth:TB10-1-31} \ldots
```

12-5-15

Getting full citations automatically

The `\cite` command automatically generates full citations if the `citefull` option is specified together with one of the following keywords: `all` (all references are full citations), `first` (first citation is full, subsequent ones are abbreviated), `chapter` (same as `first` but restarts with each chapter), and `section` (like `chapter` but restarts at the `\section` level). All settings imply `annotatorfirstsep=in`, as can be seen in the second citation in the example. If one of the above settings has been included in the configuration file and you want to turn it off for the current document, use the keyword `false`.

See Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59th edition. München, 2002 ...

As shown by Heinrichs in: Baumbach et al., § 216 the interpretation ...

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{citefull=first}
See \cite{zpo} \ldots

As shown by \cite[Heinrichs][\S\,216]{zpo}
the interpretation \ldots
```

12-5-16

`\citefullfirstfortype{`*BIBTEX-type-list*`}`

Further control is possible by specifying the BIBTEX entry types for which a full citation should be generated on the first occurrence. In the example below (otherwise similar to Example 12-5-15), we request that only entries of type `article` should be subject to this process.

For details see Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896–3207. General information can be found in Knuth: The TEXbook.

As shown by Knuth: TUGboat 10 [1989]

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{citefull=first}
\citefullfirstfortype{article}
For details see \cite{Knuth:TB10-1-31}. General
information can be found in \cite{Knuth-CT-a}.

As shown by \cite{Knuth:TB10-1-31}
```

12-5-17

`\nextciteshort{`*key-list*`}` `\nextcitefull{`*key-list*`}`
`\nextcitereset{`*key-list*`}` `\nextcitenotitle{`*key-list*`}`

Sometimes it is not correct to make the first citation to a work be the full entry, such as in an abstract or preface. On the other hand, you may want to have a certain citation show the full entry again, even though it appeared earlier. For this purpose four commands are available that modify how individual citations are presented from the given point onward.[1]

If you use `\nextciteshort`, all citations specified in the *key-list* will be typeset as short-title citations from then on (e.g., lines A, B, D in the example). If you use `\nextcitereset`, the citations will (again) be typeset in the normal way; thus, the next citation will be a full citation if there has not been one yet (lines C and F) and otherwise citations will be set as short-title citations (line E). With `\nextcitefull`, you force full entries from then on (line G). With `\nextcitenotitle`, you get only the author name(s), even if it results in ambiguous citations.

A) Knuth: The TeXbook
B) Knuth: TUGboat 10 [1989]
C) Knuth, Donald E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0
D) Knuth: TUGboat 10 [1989]
E) Knuth: The TeXbook
F) Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896–3207
G) Knuth, Donald E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0
12-5-18 H) Knuth

```
\usepackage[citefull=first]{jurabib}
\bibliographystyle{jurabib}
\nextciteshort{Knuth-CT-a,Knuth:TB10-1-31}
A) \cite{Knuth-CT-a}                      \\
B) \cite{Knuth:TB10-1-31}                 \\
\nextcitereset{Knuth-CT-a}
C) \cite{Knuth-CT-a}                      \\
D) \cite{Knuth:TB10-1-31}                 \\
\nextcitereset{Knuth-CT-a,Knuth:TB10-1-31}
E) \cite{Knuth-CT-a}                      \\
F) \cite{Knuth:TB10-1-31}                 \\
\nextcitefull{Knuth-CT-a}
\nextcitenotitle{Knuth:TB10-1-31}
G) \cite{Knuth-CT-a}                      \\
H) \cite{Knuth:TB10-1-31}
```

If full citations are used within the main document it is not absolutely necessary to assemble them in a bibliography or reference list. You may, for example, have all citations inline and use a bibliography for suggested further reading or other secondary material.

`\citeswithoutentry{`*key-list*`}`

This declaration lists those keys that should not appear in the bibliography even though they are cited in the text. The *key-list* is a list of comma-separated keys without any white space. You can repeat this command as often as necessary.

[1]The command names seem to indicate that they change the "next" citation, but in fact they change all further citations until they are overwritten.

Think of it as the opposite of \nocite. Both commands are used in the next example.

This is explained in Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29th edition. München, 2003.
As shown in Brox/Walker...

Selected further reading

Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59th edition. München, 2002

```
\usepackage{jurabib}
\renewcommand\refname
        {Selected further reading}
\bibliographystyle{jurabib}
\citeswithoutentry{aschur}
\jurabibsetup{citefull=first}
This is explained in \cite{aschur}.
\par As shown in \cite{aschur}\ldots
\nocite{zpo}
\bibliography{jura}
```
12-5-19

Suppressing the bibliography altogether

While \citeswithoutentry prevents individual works from appearing in the bibliography it is not possible to use it to suppress all entries, as you would get an empty list consisting of just the heading. If you want to omit the bibliography altogether, use \nobibliography in place of the usual \bibliography command. This command will read the .bbl file produced by BibTeX to enable citation references, but without producing a typeset result. You still need to specify jurabib as the BibTeX style and run BibTeX in the normal way.

Citations as footnotes or endnotes

All citation commands introduced so far have variants that generate footnote citations or, when used together with the endnotes package, generate endnotes. Simply prepend foot to the command name (e.g., \footcite instead of \cite, \footcitetitle instead of \citetitle, and so forth). This allows you to mix footnote and other citations freely, if needed.

The footnote citations produced by jurabib are ordinary footnotes, so you can influence their layout by loading the footmisc package, if desired.

...to use LaTeX on the web.* Also discussed by Goossens/Rahtz is generating PDF and HTML.

*Goossens, Michel/Rahtz, Sebastian: The LaTeX Web companion: integrating TeX, HTML, and XML. Reading, MA, USA: Addison-Wesley Longman, 1999, Tools and Techniques for Computer Typesetting, ISBN 0–201–43311–7.

```
\usepackage[ragged,symbol]{footmisc}
\usepackage{jurabib}
\bibliographystyle{jurabib}
\ldots to use \LaTeX{} on the
web.\footfullcite{LWC99}
Also discussed by \cite{LWC99}
is generating PDF and HTML.
```
12-5-20

Getting footnote citations automatically

If all your citations should be automatically typeset as footnotes, use the super option. In that case jurabib will automatically choose the \foot.. variants, so \cite will produce \footcite, and so forth. This is shown in the next example. There we also use citefull=first so that the first footnote looks like the one in the previous example (to save space we show only the second page, where due to the ridiculously small height of the example page the last line of that footnote is carried over). The other two citations are then automatically shortened, with the third being shortened even further because of the ibidem option (explained on the following page).

We also use the option lookat, which is responsible for the back-reference to the earlier note containing the full citation. This option is allowed only if you simultaneously use the citefull option and have all your initial citations in footnotes, as it requires a "number" to refer to.

You have to be careful to use a footnote style that produces unique numbers. If footnotes are numbered by chapter or by page, for example, then such references are ambiguous. This problem can be solved by loading the varioref package, in which case these back-references will also show page numbers. If varioref is loaded for other reasons and you do not want page references in this place, use \jbignorevarioref to suppress them. If footnotes are numbered by chapter, then an alternative solution is to use the \labelformat declaration as provided by varioref to indicate to which chapter the footnote belongs:

```
\labelformat{footnote}{\thechapter--#1}
```

The lookat option is particularly useful in combination with command \nobibliography, so that all your bibliographical information is placed in footnotes without a summary bibliography.

Also discussed is generating PDF[2] and HTML.[3]

[2]Goossens/Rahtz (as in n. 1), chap. 2.
[3]Ibid., chap. 3–4.

12-5-21

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,citefull=first,ibidem,lookat}
\ldots to use \LaTeX{} on the web.\cite{LWC99}
    \newpage % Next page shown on the left:
Also discussed is generating PDF\cite[chap.~2]
 {LWC99} and HTML.\cite[chap.~3--4]{LWC99}
```

It is possible to customize the appearance of the back-references by using the commands \lookatprefix and \lookatsuffix. Both are language dependent, which is the reason for using the \AddTo declaration (see page 733). The example sets up a style commonly seen in law citations [21].

Also discussed is generating PDF[2] and HTML.[3]

[2]Goossens/Rahtz, *supra* note 1, chap. 2.
[3]Goossens/Rahtz, *supra* note 1, chap. 3–4.

12-5-22

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,citefull=first,lookat}
\AddTo\bibsall{\renewcommand\lookatprefix
                  {, \emph{supra} note }
               \renewcommand\lookatsuffix{}}
\ldots to use \LaTeX{} on the web.\cite{LWC99}
    \newpage % Next page shown on the left:
Also discussed is generating PDF\cite[chap.~2]
 {LWC99} and HTML.\cite[chap.~3--4]{LWC99}
```

By loading the endnotes package in a set-up similar to the one from the previous example, you can turn all your citations into endnotes. As you can see, the endnotes do not have a final period added by default. If you prefer a period, add the option `dotafter` with the keyword value `endnote`.

…to typeset with graphics.[1] Also discussed is typesetting music[2] and games.[3]

Notes

[1]Goossens, Michel/Rahtz, Sebastian/Mittelbach, Frank: The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript. Reading, MA, USA: Addison-Wesley Longman, 1997, Tools and Techniques for Computer Typesetting, ISBN 0-201-85469-4

[2]Goossens/Rahtz/Mittelbach (as in n. 1), chap. 7

[3]Goossens/Rahtz/Mittelbach (as in n. 1), chap. 8

```
\usepackage{jurabib,endnotes}
\bibliographystyle{jurabib}
\jurabibsetup{citefull=first,%
              super,lookat}
\ldots to typeset with
graphics.\cite{LGC97} Also
discussed is typesetting
music\cite[chap.~7]{LGC97} and
games.\cite[chap.~8]{LGC97}
\theendnotes
```

12-5-23

Ibidem—In the same place

In some disciplines it is customary to use the Latin word "ibidem" (abbreviated as "ibid." or "ib.") if you repeat a reference to the immediately preceding citation. The `jurabib` package supports this convention in several variants if the option `ibidem` is specified. This option must be used with footnote-style citations (e.g., when using `\footcite` or with the option `super` activated).

If `ibidem` is used without a value (which is the same as using it with the keyword `strict`), then the following happens: if a citation refers to the same publication as the immediately preceding citation on the *current* page, then it is replaced by "Ibid.", if necessary keeping a *post-note*. You can see this situation in the next example: the first citation is a short-title citation; the second citation is identical so we get "Ibid." with the *post-note* dropped; and the third and forth citations refer to different parts of the same publication so we get the *post-note* as well. The fifth citation refers to a different publication by the same authors, so another short-title citation is produced. The sixth citation refers to the same publication, but the short-title citation is repeated because it is on a new page. The seventh and eighth citations are again to the other publication, so we get first a short-title citation and then "Ibid." with a *post-note*.

text[1] text[2,3] text[4,5]

[1] Brox/Walker: BSchuR, § 7.
[2] Ibid.
[3] Ibid., § 16.
[4] Ibid., § 7.
[5] Brox/Walker: Allgemeines Schuldrecht.

text[6,7] text[8]

[6] Brox/Walker: Allgemeines Schuldrecht, § 3.
[7] Brox/Walker: BSchuR.
[8] Ibid., § 15.

```
\usepackage[marginal,multiple]{footmisc}
\usepackage[super,ibidem]{jurabib}
\bibliographystyle{jurabib}
text \cite[\S\,7]{bschur}
text \cite[\S\,7]{bschur}
     \cite[\S\,16]{bschur}
text \cite[\S\,7]{bschur}
\cite{aschur}    \newpage % <---
text \cite[\S\,3]{aschur}
     \cite{bschur}
text \cite[\S\,15]{bschur}
```

12-5-24

If you typeset your document with the class option `twoside`, then you can use the keyword `strictdoublepage`. It means that "Ibid." will also be used across page boundaries as long as the preceding citation is still visible (i.e., on the same spread). Repeating Example 12-5-24 with this setting will change the sixth citation to "Ibid., § 3".

The `ibidem` option usually generates a lot of very short footnotes, so it might be economical to use it together with the `para` option of footmisc. We also add the `perpage` option so that the footnote numbers remain small. Note, however, that this makes it impossible to use the `lookat` option because the footnote numbers are no longer unique.

12-5-25

text[1] text[2,3] text[4,5]

[1] Brox/Walker: BSchuR, § 7. [2] Ibid. [3] Ibid., § 16. [4] Ibid., § 7. [5] Brox/Walker: Allgemeines Schuldrecht.

text[1,2] text[3]

[1] Ibid., § 3. [2] Brox/Walker: BSchuR. [3] Ibid., § 15.

```
\usepackage[para,multiple,perpage]{footmisc}
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=strictdoublepage}
text \cite[\S\,7]{bschur} text
\cite[\S\,7]{bschur} \cite[\S\,16]{bschur}
text \cite[\S\,7]{bschur} \cite{aschur}
\newpage          text \cite[\S\,3]{aschur}
\cite{bschur} text \cite[\S\,15]{bschur}
```

It is even possible to ignore all page boundaries by using the `nostrict` keyword. The reader might find it difficult to decipher the references, however, because "Ibid." and the citation to which it refers may be moved arbitrarily far apart. If necessary, you can disable the `ibidem` mechanism for the next citation by preceding it with `\noibidem`.

12-5-26

A page without a citation.

This page has references.[2] Or like this?[3]

[2] Ibid.
[3] Brox/Walker.

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=nostrict}
  \ldots \fullcite{bschur} \ldots
\newpage % page above not shown on the left
         A page without a citation.
\newpage This page has references.\cite{bschur}
         Or like this?   \noibidem\cite{bschur}
```

The use of "Ibid." without any further qualification allows you to reference just the immediately preceding citation. Thus, if citations are frequently mixed, the mechanism will insert short-title references most of the time. This situation will change if you use the `ibidem` option with the keyword `name` (which automatically implies `citefull=first`). In that case "Ibid." will be used with the full name of the author, thus allowing a reference to an earlier—not directly preceding—citation. If only the surnames of the authors are required, add the `authorformat` option with the keyword `reducedifibidem`. Its effect is seen in the next example, where citations to `bschur` and `zpo` alternate. A variant is to always use name and short title except for the first citation of a publication; this format can be requested with the keyword `name&title`.

If the same author is cited with more than one publication, then using the ibidem option with the name keyword is likely to produce ambiguous references. For those citations the jurabib package automatically switches to the name&title&auto method described below.

text[1] text[2,3] text[4,5] text[6]

[1] Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27th edition. München, 2002, § 7.
[2] Brox/Walker, ibid., § 8.
[3] Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59th edition. München, 2002, § 16.
[4] Brox/Walker, ibid., § 7.
[5] Baumbach et al., ibid.
[6] Baumbach et al., ibid., § 3.

```
\usepackage[marginal,ragged,multiple]{footmisc}
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=name}
\jurabibsetup{authorformat=reducedifibidem}
text \cite[\S\,7]{bschur} text
\cite[\S\,8]{bschur} \cite[\S\,16]{zpo}
text \cite[\S\,7]{bschur} \cite{zpo}
text \cite[\S\,3]{zpo}
```

12-5-27

If name&title&auto was selected (either implicitly or explicitly), then the following happens: the first citation of a publication automatically displays the full entry (citation 5 in the next example). In case of repeated citations to unambiguous works only the name of the author(s) are shown (citation 8). For ambiguous citations this will be done only for immediately following citations (citation 4). However, if there are intervening citations, then the name(s) and short titles are shown (citations 3, 6, and 7).

text[3] text[4,5] text[6,7] text[8]

[3] Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht, ibid., § 7.
[4] Brox, Hans/Walker, Wolf-Dietrich, ibid., § 8.
[5] Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59th edition. München, 2002, § 16.
[6] Brox, Hans/Walker, Wolf-Dietrich: BSchuR, ibid., § 7.
[7] Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht, ibid.
[8] Baumbach, Adolf et al., ibid., § 3.

```
\usepackage[marginal,ragged,multiple]{footmisc}
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=name&title&auto}
  Full citations: \cite{aschur} \cite{bschur}
  not shown on the left!
\newpage
  text \cite[\S\,7]{aschur} text
  \cite[\S\,8]{aschur} \cite[\S\,16]{zpo}
  text \cite[\S\,7]{bschur} \cite{aschur}
  text \cite[\S\,3]{zpo}
```

12-5-28

Another convention in certain disciplines is to replace the author's name with the Latin word "Idem" (meaning "the same") if the author of successive citations is identical. This is catered for by the option idem, which accepts the keywords strict, strictdoublepage, and nostrict with the same semantics as used with the ibidem option. Both options can be combined as shown in the next example. Due to the keywords used we get different citations: some use "Idem, ibid."; after the page break "Idem" is suppressed, because of the option strict; and in the last three citations it is used again (even with the full citation) because they all refer to different publications of Donald Knuth.

...text[1] text[2] text[3,4]...

[1]Knuth, Donald E.: The TEXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0-201-13447-0.
[2]Idem, ibid., p. 22.
[3]Leunen, Mary-Claire van: A handbook for scholars. Walton Street, Oxford OX2 6DP, UK: Oxford University Press, 92.
[4]Idem, ibid.

...text[5] text[6] text[7] text[8,9]...

[5]Leunen, Mary-Claire van, ibid.
[6]Idem, ibid., p. 16.
[7]Knuth, Donald E.: The TEXbook, ibid., p. 308.
[8]Idem: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896-3207.
[9]Idem: The TEXbook, ibid., p. 80.

12-5-29

```
\usepackage[flushmargin,%
            multiple]{footmisc}
\usepackage[super,idem=strict,%
            ibidem=name]{jurabib}
\bibliographystyle{jurabib}
\ldots text \cite{Knuth-CT-a}
text \cite[p.~22]{Knuth-CT-a}
text \cite{vLeunen:92}
\cite{vLeunen:92}\ldots
\newpage % <--
\ldots text \cite{vLeunen:92}
text \cite[p.~16]{vLeunen:92}
text \cite[p.~308]{Knuth-CT-a}
text \cite{Knuth:TB10-1-31}
\cite[p.~80]{Knuth-CT-a}\ldots
```

You have to ask yourself whether this type of citation is actually helpful to your readers. Butcher [29], for example, argues against it. Of course, you may not have a choice in the matter—it might be required. You should, however, note that two citations in the previous example are actually wrong: van Leunen is a female author, so the correct Latin form would be "Eadem" and not "Idem" (though some style manuals do not make that distinction). If necessary, jurabib offers possibilities for adjusting your citations even on that level of detail; see page 734.

There is another convention related to recurring citations, though it is becoming less common: to signal that a citation refers to an earlier reference, it is flagged with *op. cit.* (*opere citato*, "in the work cited"). This practice is supported with the option `opcit`. The citation should be "close by" so that the reader has a chance to find it. For this reason jurabib offers the keywords `chapter` and `section` in analogy to the `citefull` option.

...text[1] text[2] text[3] some more text[4,5]

[1]Knuth, Donald E.: The TEXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0-201-13447-0.
[2]Idem, *op. cit.*, p. 22.
[3]Free Software Foundation: GNU Make, A Program for Directing Recompilation. 2000.
[4]Knuth, *op. cit.*
[5]Free Software Foundation, *op. cit.*

12-5-30

```
\usepackage[multiple]{footmisc}
\usepackage[super,idem=strict,%
    citefull=first,opcit]{jurabib}
\bibliographystyle{jurabib}
\ldots text \cite{Knuth-CT-a} text
\cite[p.~22]{Knuth-CT-a} text
\cite{GNUMake} some more text
\cite{Knuth-CT-a}\cite{GNUMake}
```

In law citations [21], it is common to use the word "*supra*" to indicate a reference to a previous citation. This can be accomplished by changing the `\opcit` command, which holds the generated string, as follows:

```
\renewcommand\opcit{\textit{supra}}
```

Alternatively, you can use the method shown in Example 12-5-22 on page 727.

Cross-referencing citations

BIBTEX supports the notion of cross-references between bibliographical entries via the `crossref` field. For example, an entry of type `inproceedings` can reference the proceedings issue in which it appears. Depending on the number of references to such an issue, BIBTEX then decides whether to produce a separate entry for the issue or to include information about it in each `inproceedings` entry. See Section 13.2.5 for details.

If BIBTEX decides to produce separate entries for the cross-referenced citations, a question arises about what should happen if they are referenced in a `\fullcite` or `\footfullcite` command in the text. To handle this situation jurabib offers three keywords applicable to the `crossref` option: with the keyword `normal` (the default), cross-references are typeset as an `author/editor`, `title` combination (or `shortauthor`, `shorttitle` if available); with the keyword `short`, only the `author` or `editor` is used as long as there are no ambiguities; and with the keyword `long`, cross-references are listed in full. The default behavior is shown below (where the editors and the short title were selected by jurabib).

Mittelbach, Frank/Rowley, Chris: The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In Vanoirbeek/Coray: EP92

Southall, Richard: Presentation Rules and Rules of Composition in the Formatting of Complex Text. In Vanoirbeek/Coray: EP92

Mittelbach/Rowley

```
\usepackage{jurabib}
\jurabibsetup{citefull=first,
              crossref=normal}
\bibliographystyle{jurabib}
\cite{MR-PQ}       \par
\cite{Southall}    \par
\cite{MR-PQ}
```

12-5-31

You can combine any of the three keywords with the keyword `dynamic`, in which case the first cross-reference is given in a longer form when cited the first time and in the shorter form on all later occasions. Here we combine it with the keyword `long` so that we get a full citation to Vanoirbeek/Coray in the first citation and a short title citation in the second.

Frank Mittelbach/Chris Rowley: The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In Christine Vanoirbeek/Giovanni Coray, editors: EP92—Proceedings of Electronic Publishing, '92. Cambridge: Cambridge University Press, 1992

Richard Southall: Presentation Rules and Rules of Composition in the Formatting of Complex Text. In Vanoirbeek/Coray: EP92

```
\usepackage{jurabib}
\jurabibsetup{citefull=first,
    authorformat=
           citationreversed,
    crossref={dynamic,long}}
\bibliographystyle{jurabib}
\cite{MR-PQ}       \par
\cite{Southall}
```

12-5-32

Author-date citation support

As mentioned earlier, jurabib supports the commands `\citet` and `\citep` as introduced by natbib. It also offers `\citealt`, `\citealp`, `\citeauthor`, `\citeyear`, and `\citeyearpar`. Those forms for which it makes sense are also available as footnote citations by prefixing the command name with foot (e.g., \footcitet). Not provided are the starred forms available with natbib.

12-5-33

Goossens/Rahtz (1999)
Goossens/Rahtz (1999, chap. 2)
see Goossens/Rahtz (1999, chap. 2)
pre-note only: see Goossens/Rahtz (1999)

(Goossens/Rahtz, 1999)
(Goossens/Rahtz, 1999, chap. 2)
(see Goossens/Rahtz, 1999, chap. 2)
pre-note only: (see Goossens/Rahtz, 1999)

Knuth, 1986
Knuth
(1986)

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\citet{LWC99}                          \\
\citet[chap.~2]{LWC99}                 \\
\citet[see][chap.~2]{LWC99}            \\
pre-note only: \citet[see][]{LWC99} \\[5pt]
\citep{LWC99}                          \\
\citep[chap.~2]{LWC99}                 \\
\citep[see][chap.~2]{LWC99}            \\
pre-note only: \citep[see][]{LWC99} \\[5pt]
\citealp{Knuth-CT-a}                   \\
\citeauthor{Knuth-CT-a}                \\
\citeyearpar{Knuth-CT-a}
```

A combination of author-date and short-title citations is achieved by setting authorformat=year, as already introduced in Example 12-5-5. The formatting of the year can be influenced with \jbcitationyearformat, and the position of the date can be moved after the title (if present) by specifying \jbyearaftertitle.

12-5-34

Brox/Walker 2003
Otto Palandt/*Heinrichs* 2003, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat=year,annotatorformat=italic}
\renewcommand\jbcitationyearformat[1]{\oldstylenums{#1}}
\jbyearaftertitle
\cite{aschur}  \\  \cite[Heinrichs][\S\,26]{bgb}
```

Language support

Most strings that are generated automatically in a bibliography entry or as part of a full citation, are language dependent; they depend on the main language of the document. The jurabib package supports this by collaborating with the babel package. Depending on the main language of the document (determined by the last option to the babel package), jurabib loads a special language definition file (extension .ldf) that contains definitions for all kinds of commands that produce textual material within citations and bibliography entries. At the moment approximately 10 languages are supported. These language files (e.g., enjbbib.ldf for English) are a good source for finding out details about customization possibilities. To modify such a command from such files for a particular language (or for all languages), jurabib offers the \AddTo declaration.

\AddTo\bibsall{*code*}   \AddTo\bibs⟨*language*⟩{*code*}

The declaration \AddTo takes two arguments: a command name that holds all language-related definitions for one language and the *code* that should be added to this storage place.[1] The first argument is either `\bibsall`, in which case *code* is used for all languages, or `\bibs⟨language⟩` (e.g., `\bibgerman`), in which case *code* is applied for that particular *language*.[2] In Example 12-5-14 on page 723 and Example 12-5-22 on page 727 we used `\AddTo` to change the presentation of back-references for all languages, by adding the redefinitions to `\bibsall`. Below we shorten the "Ibid." string when typesetting in the English language. The default for other languages is left unchanged in this case.

Some text[1] and[2] or[3] and more text.[4]

[1]van Leunen: A handbook for scholars.
[2]Ib.
[3]Knuth, Donald E.: The TEXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0.
[4]Knuth, Donald E., ib.

```
\usepackage[super,ibidem,titleformat=all]{jurabib}
\AddTo\bibsenglish{\renewcommand\ibidemname{Ib.}%
                   \renewcommand\ibidemmidname{ib.}}
\bibliographystyle{jurabib}
Some text\cite{vLeunen:92} and\cite{vLeunen:92}
\jurabibsetup{ibidem=name} % <-- change convention
or\cite{Knuth-CT-a} and more text.\cite{Knuth-CT-a}
```

12-5-35

While certain strings—calling an editor (`\editorname`) "(Hrsg.)", for example—should clearly be consistent throughout the whole bibliography, certain other aspects—most importantly, hyphenation—depend on the language used in the actual entry. For instance, a book with a German title should be hyphenated with German hyphenation patterns, regardless of the main language of the document. This is supported by jurabib through an extra field (`language`) in the BIBTEX database file. If that field is specified in a given entry, then jurabib assumes that the title should be set in that particular language. Thus, if hyphenation patterns for that language are available (i.e., loaded in the format), they will be applied. For instance, if we repeat the last part of Example 12-5-6 from page 719 with babel loaded, we get the correction hyphenation:

Allgemeines Schuldrecht; Besonderes Schuld-
recht

```
\usepackage[ngerman,english]{babel}
\usepackage{jurabib} \bibliographystyle{jurabib}
\citefield{title}{aschur,bschur}
```

12-5-36

Distinguishing the author's gender

Earlier, we mentioned that the female form of "Idem" is "Eadem". In the German language, we have "Derselbe" (male), "Dieselbe" (female), "Dasselbe" (neuter), and "Dieselben" (plural). To be able to distinguish the gender of the author, jurabib offers the BIBTEX field `gender`, which takes a two-letter abbreviation for the gender as its value.

[1]The babel package uses a similar mechanism with the `\addto` declaration.

[2]Unfortunately, jurabib does not use exactly the same concept as babel. If you specify `ngerman` with babel to get German with new hyphenation patterns, then this is mapped to `german`, so you have to update `\bibsgerman`. If you use any of the dialects (e.g., `austrian`), then jurabib will not recognize those and will use `english` after issuing a warning. In that case use `\bibsall` for changing definitions.

gender	*Meaning*	*In Citation*	*In Bibliography*
sf	single female	\idemSfname, \idemsfname	\bibidemSfname, \bibidemsfname
sm	single male	\idemSmname, \idemsmname	\bibidemSmname, \bibidemsmname
pf	plural female	\idemPfname, \idempfname	\bibidemPfname, \bibidempfname
pm	plural male	\idemPmname, \idempmname	\bibidemPmname, \bibidempmname
sn	single neuter	\idemSnname, \idemsnname	\bibidemSnname, \bibidemsnname
pn	plural neuter	\idemPnname, \idempnname	\bibidemPnname, \bibidempnname

Table 12.1: Gender specification in jurabib

Possible values and the commands that contain the "Idem" strings, if specified, are given in Table 12.1. The commands with an uppercase letter in their name are used at the beginning of a sentence, the others in mid-sentence. Those starting with `\bibidem..` are used in the bibliography if the option `bibformat` with the keyword `ibidem` is specified. Since the feature is computing intensive, it is not activated by default but has to be requested explicitly. Thus, to change to "Eadem" in case of female authors, we have to specify values for `\idemSfname` and `\idemsfname` and use the option `lookforgender`.

Some text[1] and[2] or[3] and more text.[4]

[1]van Leunen: A handbook for scholars.
[2]Eadem: A handbook for scholars.
[3]Knuth: The TeXbook.
[4]Idem: The TeXbook.

12-5-37

```
\usepackage[super,idem=strict,titleformat=all,
                lookforgender=true]{jurabib}
\AddTo\bibsenglish{\renewcommand\idemSfname{Eadem}%
                   \renewcommand\idemsfname{eadem}}
\bibliographystyle{jurabib}
Some text\cite{vLeunen:92} and\cite{vLeunen:92}
or\cite{Knuth-CT-a} and more text.\cite{Knuth-CT-a}
```

Customizing the in-text citation layout further

Most of the author and title formatting is handled by the options `authorformat` and `titleformat`, which were discussed earlier. There also exist a few more options and commands that we have not mentioned so far.

If the whole citation should be surrounded by parentheses, simply specify the option `round` or `square`.

To place information about the edition as a superscript after the short title, specify the option `superscriptedition`. With a value of `all` this will be applied to all short-title citations, with the keyword `commented` applying only to publications of type `commented`, and with the keyword `multiple` applying only to publications that are cited with several different editions. The last two options are primarily intended for juridical works.

[Baumbach et al.: ZPO[59]]
[Brox/Walker[27], § 3]
[Otto Palandt/Heinrichs[62]]

12-5-38

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{square,superscriptedition={all}}
\citetitle{zpo}\\ \cite[\S\,3]{bschur}\\ \cite[Heinrichs][]{bgb}
```

Alternatively, you can explicitly specify in the BIBTEX database for each entry whether the edition should be shown as a superscript by setting the special field `ssedition` to the value 1 and by using the option `superscriptedition` with the keyword `switch`.

By specifying `authorformat=and` you will get author names separated by commas and "and" (actually by `\andname`, a command that has different values in different languages). But you cannot have the second and third author names separated by ", and" in this way. For adjustments on such a fine level, you can redefine `\jbbtasep` (between two authors separation), `\jbbfsasep` (between first and second authors separation), and `\jbbstasep` (between second and third authors separation).[1]

```
\usepackage[round]{jurabib}
\renewcommand\jbbtasep{ and }   \renewcommand\jbbfsasep{, }
\renewcommand\jbbstasep{, and } \bibliographystyle{jurabib}
\cite{aschur} \\ \cite{LGC97}
```

(Brox and Walker)
(Goossens, Rahtz, and Mittelbach)

12-5-39

You may also want to manually specify the fonts used for the author names and the short title, instead of relying on the possibilities offered by the supplied options. For this you have `\jbauthorfont`, `\jbannotatorfont`, `\jbactualauthorfont`, `\jbauthorfontifannotator`, and `\jbtitlefont` at your disposal, all of which are commands with one argument.

Customizing the bibliography layout

The formatting of the bibliography in standard LaTeX or with natbib is largely controlled by the used BIBTEX style file or, if the bibliography entries are manually produced, by the formatting directives entered by the user. For example, a citation to the entry `Knuth-CT-a` from our sample database would be formatted by natbib's `plainnat` as follows:

```
Donald~E. Knuth.
\newblock {\em The {\TeX}book}, volume~A of {\em Computers and Typesetting}.
\newblock Ad{\-d}i{\-s}on-Wes{\-l}ey, Reading, MA, USA, 1986.
```

This means that formatting decisions, such as using emphasis for the title of the book and the series, and the presentation of the "volume" field, have all been made by the BIBTEX style file.

In contrast, the BIBTEX styles that come with the jurabib package use a drastically different approach: their output is highly structured, consisting of a large number of LaTeX commands, so that the final formatting (as well as the order of elements to some extent) can still be tweaked on the LaTeX level. In fact, they have to be adjusted on that level if you are not satisfied with the formatting produced

[1] No other possibilities are needed, since jurabib always uses "et al." whenever there are four or more authors.

from their default definitions. For example, the same citation as above processed with the jurabib BibTeX style results in the following entry:

```
\jbbibargs {\bibnf {Knuth} {Donald~E.} {D.~E.} {} {}} {Donald~E. Knuth} {au}
 {\bibtfont {The {\TeX}book}\bibatsep\ \volumeformat {A} Computers and
 Typesetting\bibatsep\ \apyformat {Reading, MA, USA\bpubaddr {}
 Ad{\-d}i{\-s}on-Wes{\-l}ey\bibbdsep {} 1986} \jbPages{ix + 483}\jbisbn {
 0--201--13447--0}} {\bibhowcited} \jbdoitem \bibAnnoteFile {Knuth-CT-a}
```

Most of the above commands are further structured. The `\bibnf` command takes five arguments (the different parts of the author's name) and, depending on which are nonempty, passes them on to commands like `\jbnfIndNoVonNoJr` (name without "von" and "Junior" parts) for further processing. Consequently, it is possible to interact with this process at many levels so that all kinds of requirements can be catered for, although this somewhat complicates the customization of the layout. For this reason we restrict ourselves to showing just the most important customization possibilities. For further control strategies, consult the package documentation.

In the default set-up, the formatting of the bibliography is fairly independent of that used for the citations. If you specify `authorformat=italic`, author names are typeset in italics in the text but there is no change in the bibliography. The easiest way to change that is to use the option `biblikecite`; then formatting decisions for the citations will also be used in the bibliography as far as possible. If that is not desired or not sufficient, explicit formatting directives are available; they are discussed below.

The fonts used in a bibliographical entry are controlled by the following set of commands: `\biblnfont` and `\bibfnfont` for formatting the last and first names of the author, and `\bibelnfont` and `\bibefnfont` for the last and first names of the editor, if present. The command `\bibtfont` is used for titles of books, `\bibbtfont` for titles of essays (i.e., entries involving a BibTeX `booktitle` field), and `\bibjtfont` for titles, or rather names, of journals. The font for article titles within such a journal is customized with `\bibapifont`. The commands all receive the text they act upon as an argument, so any redefinition must also use an argument or `\text..` font commands as shown in the next example (picking the argument up implicitly).

KNUTH, DONALD E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

KNUTH, DONALD E.: *"Typesetting Concrete Mathematics"*. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207

12-5-40

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\renewcommand\biblnfont{\MakeUppercase}
\renewcommand\bibfnfont{\textsc}
\renewcommand\bibtfont {\textsf}
\renewcommand\bibapifont[1]{\textit{``#1''}}
\nocite{Knuth-CT-a,Knuth:TB10-1-31}
\bibliography{tex}
```

The punctuation separating different parts in the entry can be customized by another set of commands: \bibansep sets the punctuation and space after the author name, \bibeansep does the same after the editor name, \bibatsep produces punctuation after the title (the space is already supplied!), and \bibbdsep is the punctuation *before* the date. With \bibjtsep the journal title separation is set. There are similar commands for adjusting other parts.[1] In the next example we use these commands to remove the default colon after the author's name and then typeset a semicolon after the title, no comma before the year, and the word "in" before the journal name. We also use the dotafter option with the keyword bibentry to add a final period after each entry.

Knuth, Donald E. Typesetting Concrete Mathematics; in TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207.

Mittelbach, Frank/Rowley, Chris The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In **Vanoirbeek/Coray** EP92, 261–273.

Vanoirbeek, Christine/Coray, Giovanni, editors EP92— Proceedings of Electronic Publishing, '92; Cambridge: Cambridge University Press 1992.

```
\usepackage[dotafter=bibentry]
           {jurabib}
\bibliographystyle{jurabib}
\renewcommand\bibjtsep{in }
\renewcommand\bibansep{ }
\renewcommand\bibatsep{;}
\renewcommand\bibbdsep{}
\nocite{Knuth:TB10-1-31,MR-PQ}
\bibliography{tex}
```

12-5-41

We already saw that the separation between different author names in a citation can be adjusted by means of the authorformat option and various keywords. However, except for the keyword allreversed, this has no effect on the entries in the bibliography. To modify the formatting there, you have to redefine the commands \bibbtasep, \bibbfsasep, and \bibbstasep. The naming convention is the same as for the corresponding citation commands. A similar set of commands, \bibbtesep, \bibbfsesep, and \bibbstesep, is available to specify the separation between editor names in an entry.

Hans Brox and **Wolf-Dietrich Walker:** Allgemeines Schuldrecht. 29th edition. München, 2003

Michel Goossens, **Sebastian Rahtz**, and **Frank Mittelbach:** The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript. Reading, MA, USA: Addison-Wesley Longman, 1997, Tools and Techniques for Computer Typesetting, xxi + 554, ISBN 0–201–85469–4

```
\usepackage[authorformat=allreversed]
           {jurabib}
\bibliographystyle{jurabib}
\renewcommand\bibbtasep{ and }
\renewcommand\bibbfsasep{, }
\renewcommand\bibbstasep{, and }
\nocite{aschur,LGC97}
\bibliography{tex,jura}
```

12-5-42

Adjusting the general layout of the bibliography

The main option for influencing the general layout of the bibliography list is bibformat, which can take a number of keywords as its value. If you specify the keyword nohang, then the default indentation (of 2.5em) for the second and

[1]This area of jurabib is somewhat inconsistent in its naming conventions and command behavior. Perhaps this will change one day.

subsequent lines of a bibliographical entry is suppressed. Alternatively, you can explicitly set the indentation by changing the dimension parameter `\jbbibhang`, as in the next example. There we also use the keywords `compress` (using less space around entries) and `raggedright` (typesetting entries unjustified). For improved quality, especially when typesetting to a small measure, you may want to load the package ragged2e. Note the use of the `newcommands` option to overload the standard `\raggedright` (as used by jurabib) with `\RaggedRight`.

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29th edition. München, 2003

Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59th edition. München, 2002

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27th edition. München, 2002

```
\usepackage[newcommands]{ragged2e}
\usepackage[bibformat={compress,%
                       raggedright}]
           {jurabib}
\bibliographystyle{jurunsrt}
\setlength\jbbibhang{1pc}
\nocite{aschur,zpo,bschur}
\bibliography{jura}
```

12-5-43

If you use the keyword `tabular`, then the bibliography is set in a two-column table with the left column containing the author(s) and the right column the remainder of the entry. By default, the first column is one third of `\textwidth` and both columns are set ragged. The defaults can be changed by redefining a number of commands, as shown in the next example. The width of the right column is specified by

```
\renewcommand\bibrightcolumn{\textwidth-\bibleftcolumn-\bibcolumnsep}
```

Normally it is enough to change `\bibleftcolumn` and/or `\bibcolumnsep`. The calc package is automatically loaded by jurabib, so we can make use of it when specifying dimensions.

Brox, Hans/ Walker, Wolf-Dietrich	Allgemeines Schuldrecht. 29th edition. München, 2003
Knuth, Donald E.	Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207
Free Software Foundation	GNU Make, A Program for Directing Recompilation. 2000

```
\usepackage[bibformat=tabular]{jurabib}
\bibliographystyle{jurabib}
\renewcommand\bibleftcolumn{6.5pc}
\renewcommand\bibcolumnsep{1pc}
\renewcommand\bibleftcolumnadjust
                       {\raggedright}
\renewcommand\bibrightcolumnadjust{}
\nocite{aschur,Knuth:TB10-1-31}
\nocite{GNUmake}
\bibliography{tex,jura}
```

12-5-44

If you use the keyword `numbered`, the bibliography will be numbered even though the actual citations in the text use the author-date or short-title scheme. Currently, it is impossible to refer to those numbers.

Some publishers' house styles omit the author's name (or replace it by a dash or other character) if that author is cited with several works. This is supported through the keyword `ibidem`, which by default generates "Idem" or, more precisely, the result from executing `\bibidemSmname`. To get a (predefined) rule instead, use `\jbuseidemhrule`. If you want something else, redefine `\bibauthormultiple`. Both possibilities are shown in the next example. The jurabib package automatically detects if an entry appears on the top of a page and will use the author name in that case. Because of this mechanism it may take several (extra) LaTeX runs before the document compiles without "Rerun to get..."

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27th edition. München, 2002

—— Allgemeines Schuldrecht. 29th edition. München, 2003

Knuth, Donald E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

—— Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207

```
\usepackage[bibformat=ibidem]
          {jurabib}
\bibliographystyle{jurabib}
\jbuseidemhrule  % use default rule
% Alternative generic redefinition
% instead of the default rule:
%\renewcommand\bibauthormultiple
%            {[same name symbol]}
\nocite{aschur,bschur}
\nocite{Knuth-CT-a,Knuth:TB10-1-31}
\bibliography{tex,jura}
```

12-5-47

A variant bibliography layout collecting works under the author names is available through the keyword `ibidemalt`. This keyword automatically implies the keyword `compress`.

Baumbach, Adolf et al.:

▷ Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59th edition. München, 2002

Brox, Hans/Walker, Wolf-Dietrich:

▷ Besonderes Schuldrecht. 27th edition. München, 2002

▷ Allgemeines Schuldrecht. 29th edition. München, 2003

Palandt, Otto:

▷ Bürgerliches Gesetzbuch. 62th edition. München: Beck Juristischer Verlag, 2003

```
\usepackage{jurabib}
\jurabibsetup{bibformat=ibidemalt}
\bibliographystyle{jurabib}
\nocite{aschur,bschur,zpo,bgb}
\bibliography{jura}
```

12-5-

Annotated bibliographies

If you want to produce an annotated bibliography, use the option `annote`. If the current BibTeX entry has an `annote` field, it will be typeset after the entry using `\jbannoteformat` to format it (the default is to typeset it in `\small`). If there is no `annote` field, then jurabib searches for a file with the extension `.tex` and the key of the entry as its base name. If this file exists, its contents will be used as the annotation text.

Knuth, Donald E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

The authoritative user manual on the program TeX by its creator.

12-5-47

```
\begin{filecontents}{Knuth-CT-a.tex}
  The authoritative user manual on the program \TeX{}
  by its creator.
\end{filecontents}
\usepackage[annote]{jurabib}\bibliographystyle{jurabib}
\renewcommand\jbannoteformat[1]
  {{\footnotesize\begin{quote}#1\end{quote}}}
\nocite{Knuth-CT-a}
\bibliography{tex}
```

Since it is a nuisance to have many files (one for each annotation) cluttering your current directory, jurabib offers a search path declaration in analogy to the `\graphicspath` command provided by the graphics package. Thus, after

```
\bibAnnotePath{{./books}{./articles}}
```

annotation files are searched for in the subdirectories `books` and `articles` of the current directory.

Using external configuration files

Customization of jurabib is possible on two levels: by specifying options or, for finer control, by redefining certain declarations or executing commands. In the previous sections we have already encountered a number of package options together with the keywords they accept but they represented less than a third of what is available. In the default configuration file `jurabib.cfg`, you will find a `\jurabibsetup` declaration listing *all* options together with all their keyword values—nearly 100 possibilities in total. They are all commented out so that you can produce your own configuration file by copying the default one and uncommenting those options you want to execute normally. If you save this configuration in a file with extension `.cfg`, you can load it instead of the default configuration by using the `config` option. For example,

```
\usepackage[config=law]{jurabib}
```

will load the option file `law.cfg`, which should contain a `\jurabibsetup` declaration and possibly some additional customization commands. For example, such a file might contain

```
\jurabibsetup{lookat,opcit,commabeforerest,titleformat=colonsep}
\renewcommand\opcit{\textit{supra}}
```

and perhaps some other initializations to implement citations for juridical publications. As mentioned earlier, such defaults stored in a file can be overwritten by using additional options during loading or with a `\jurabibsetup` declaration in the preamble.

BIBTEX styles for jurabib

The jurabib package is distributed together with four BIBTEX style files: `jurabib`, `jureco`, `jurunsrt`, and `jox`. They differ only in minor details: `jureco` produces a slightly more compact bibliography, leaving out some data, while `jurunsrt` is the same as `jurabib` without sorting, so that the references appear in order of their citation in the document. The `jox` style produces references in "Oxford style". Since jurabib requires very specially formatted `\bibitem` commands, the above styles are currently the only ones that can be used together with the package.

All four styles provide a number of additional BIBTEX entries as well as a number of additional fields for existing entries. Having additional fields in a BIBTEX database is usually not a problem, since BIBTEX ignores any field it doesn't know about. Thus, such a database can be used with other BIBTEX styles that do not provide these fields. Additional entries are slightly different, since using them means you have to use jurabib to be able to refer to them.

Additional BIBTEX types

The additional entries are `www` for citing a URL, `periodical` for periodicals that are not cited by year but by volume number, and `commented` for commentaries in juridical works.

The standard BIBTEX fields are described in Table 13.2 on page 765. The following additional fields are available when using one of the jurabib BIBTEX styles:

`annote` An annotation that is typeset if jurabib is used with the option `annote`; see page 740 for details.

`booktitleaddon` Extra information to be typeset after a `booktitle` text of a collection.

`dissyear` Year of a dissertation, habilitation, or other source if that work is also being published as a book (perhaps with a different year).

`editortype` Position of the person mentioned in the `editor` field (if not really an "editor").

`flanguage` Foreign language, in case of a translated work.

`founder` In juridical works, the original founder of a publication (in contrast to the editor). The name is shown followed by the replacement text of `\foundername`, which defaults to "`␣(Begr.)`".

`gender` Gender of the author or authors. The jurabib package uses this information to select the right kind of words for "Idem" in the current language; see page 734.

`howcited` Text to use for back-reference information, or `1` to indicate that a normal back-reference should be generated. This field is evaluated by the option `howcited` if used together with the keyword `normal`; see page 721.

`oaddress/opublisher/oyear` Information about the first edition of a work.

`shortauthor` Text to use as the author information in a short-title citation. By default, jurabib automatically selects the last name (or names) from the author or editor field.

`shorttitle` Text to use as the title information in a short-title citation. If it is not specified the whole `title` is used.

`sortkey` String to be used for sorting in unusual situations. To sort "von Bismarck, Otto" under B, you can use `sortkey="Bismarck, Otto von"`.

`ssedition` Flag to indicate that this entry should be typeset with the edition shown as a superscript. It requires the use of the `superscriptedition` option together with the keyword `switch`; see page 735.

`titleaddon` Extra information to be placed after a title but not used, for example, when generating a short title.

`totalpages` Total number of pages in a publication. If present, it will be shown followed by the replacement text of the command `\bibtotalpagesname`, which is language dependent.

`translator` Translator of the publication.

`updated` Date of the last update in a loose-leaf edition or a similar work. The field is only available for the BIBTEX type `commented`. By default, "last update *date*" is generated. This can be customized through the commands `\updatename` and `\updatesep`.

`urldate` Date when a URL was known to be current. By default, jurabib produces the string "visited on *date*" when this field is used. It can be changed by redefining the command `\urldatecomment`.

`url` A URL related to the current publication. In case of the entry type `www`, it is required; otherwise, it is optional.

`volumetitle` A volume title that follows the volume number in the presentation. This field is available for the types `book`, `commented`, `incollection`, and `inbook`.

12.5.2 camel—Dedicated law support

Anyone who needs to comply with the conventions used in (Anglo-American) legal works may also be interested in the camel "bibliography engine" [15, 16] written by Frank Bennett, Jr., in 1997. It implements citation conventions as specified in the *Blue Book* [21] (though for an earlier edition) and offers features such as classified citations. It can be used to generate table of cases, statutes, and much more. However, as camel is currently not being developed any further (volunteers welcome), one has to take some rough edges in the software as features.

In contrast to the packages described so far, camel uses its own set of commands to specify citations (`\source` instead of `\cite`), bibliographical databases (`\citationdata` instead of `\bibliography`), citation conventions (`\citationstyle` instead of `\bibliographystyle`), and printed bibliographies (`\printbibliography` as the second part of the functionality of `\bibliography`).

The next example shows these commands in action. The `\source` command takes an optional first argument in which one can specify what kind of citation should be given (e.g., "`f`" for full reference, "`t`" for title omitted, "`a`" for author name omitted). A second optional argument after the *key* can be used to specify page numbers in the reference.

An interesting feature is that the package recognizes so-called interword connectors between citations (e.g., "`see-also`" and "`cited-in`" in our example). As a result those citations are considered to belong together and are automatically placed into the same footnote.

. . . text [1] . . . somewhat later . . . [2]

References

D. E. KNUTH, THE TEXBOOK (Computers and Typesetting, 1986).

[1] D. E. KNUTH, (Computers and Typesetting, 1986); *see also* Knuth, TUGBOAT, V. 10, N. 1, p. 31 (1989).

[2] H. BROX AND W.-D. WALKER, BESONDERES SCHULDRECHT 24, 130, 216 (27. ed. 2002) *cited in* ZIVILPROZESSORDNUNG MIT GERICHTSVERFASSUNGSGESETZ UND ANDEREN NEBENGESETZEN (59. neubearb. ed. 2002).

```
\usepackage{camel}
\forcefootnotes
\citationstyle{law}
\citationdata{jura,tex}

\ldots text \source[t]{Knuth-CT-a}
see-also \source[f]{Knuth:TB10-1-31}
\ldots\ somewhat later \ldots
\source[f]{bschur}[24,130,216]
cited-in \source[a]{zpo}

\printbibliography[labels=false]{all}
```

12-5-48

Another feature that can be of interest is the ability to produce subject bibliographies using the `\citationsubject` declaration.

. . . text [1] . . . later. . . [2]

Law

[1] H. BROX AND W.-D. WALKER, BESONDERES SCHULDRECHT (27. ed. 2002)

TEX literature

[1] D. E. KNUTH, THE TEXBOOK (Computers and Typesetting, 1986)

[2] Knuth, *Typesetting Concrete Mathematics*, TUGBOAT, V. 10, N. 1, p. 31 (1989)

[1] THE TEXBOOK (Computers and Typesetting, 1986); *see also Typesetting Concrete Mathematics*, TUGBOAT, V. 10, N. 1, p. 31 (1989).

[2] H. BROX AND W.-D. WALKER, (27. ed. 2002).

```
\usepackage{camel}
\citationsubject[o=tts,i=ttb]
     {tex}{\TeX{} literature}
\citationsubject[o=lts,i=ltb]
          {jur}{Law}
\forcefootnotes
\citationstyle{law}
\citationdata{jura,tex}

\ldots text
\source[a,s=tex]{Knuth-CT-a}
see-also \source[f,s=tex]
          {Knuth:TB10-1-31}
\ldots later\ldots
\source[t,s=jur]{bschur}
\printbibliography{jur}
\printbibliography{tex}
```

12-5-49

The citation data are written to external files (extension specified with `o=` on the `\citationsubject` declaration). Such files have to be processed by *MakeIndex*:

```
makeindex -s camel.ist -o ⟨jobname⟩.ttb ⟨jobname⟩.tts
makeindex -s camel.ist -o ⟨jobname⟩.ltb ⟨jobname⟩.lts
```

The results are then read back in (`i=` argument) on the next LaTeX run.

12.6 Multiple bibliographies in one document

In large documents that contain several independent sections, such as conference proceedings with many different articles, or in a book with separate parts written by different authors, it is sometimes necessary to have separate bibliographies for each of the units. In such a scenario citations are confined to a certain part of the document, the one to which the bibliography list belongs.

A complementary request is to have several bibliographies in parallel, such as one for primary sources and one for secondary literature. In that case one has to be able to reference works in different bibliographies from any point in the document.

Both requests can be automatically resolved if none of the bibliographies contain the same publication[1] and you are prepared to produce the bibliographies manually, by means of several `thebibliography` environments without using BIBTeX. In that case the `\bibitem` commands within the environment provide the right cross-referencing information for the `\cite` commands (or their variants) to pick up from anywhere in the document. Having the same publication in several bibliographies (or more exactly the same reference key) is not possible, since that would lead to a "multiply defined labels" warning (see page 928) and to incorrect references. Of course, this could be manually corrected by choosing a different key for such problematical citations.

Being deprived of using BIBTeX has a number of consequences. First, it will be more difficult to impose a uniform format on the bibliographical entries (something that BIBTeX automatically handles for you). Second, using an author-date or short-title citation scheme will be difficult (since natbib requires a special structure within the optional argument of `\bibitem`) to downright impossible (since the structure required by jurabib is not suitable for manual production); see Section 12.3 for a discussion of the required `\bibitem` structures in both cases.

To be able to use BIBTeX for this task people had to find a way to generate several `.bbl` files from one source document. As discussed in Section 12.1.3, the interaction with BIBTeX normally works as follows: each citation command (e.g., `\cite`) writes its *key-list* as a `\citation` command into the `.aux` file. Similarly, `\bibliography` and `\bibliographystyle` commands simply copy their arguments to the `.aux` file. BIBTeX then reads the master `.aux` file (and, if necessary,

[1]This could happen, for example, if you compile the proceedings of a conference and each article therein has its own bibliography.

	Bibliographies per Unit			
	chapterbib	bibunits	bibtopic	multibib
Bibliography per chapter	x	x	x	n/a
Bibliography per other unit	Restrictions	x	x	n/a
Deal with escaping citations	x	Restrictions	Error	n/a
Additional global bibliography	Labor	x	No	n/a
Group bibliographies together	x	No	No	n/a
Multiple global bibliographies	No	No	x	x
Multiple bibliographies per unit	No	No	x	No
cite compatible	x	x	x	x
jurabib compatible	x	x	Restrictions	x
natbib compatible	x	x	x	x
Support for unsorted BIBTEX styles	x	x	No	x
Works with standard .bib files	x	x	No	x
	chapterbib	bibunits	bibtopic	multibib
			Per Topic	

Blue entries indicate features (or missing features) that may force a selection.

Table 12.2: Comparison of packages for multiple bibliographies

those from \included files) searching for occurrences of the above commands. From the provided information it produces a single .bbl file. To make BIBTEX work for the above scenarios, four problems have to be solved:

1. Generate one .aux file for every bibliography in the document that can be used as input for BIBTEX.
2. Ensure that each citation command writes its information to the correct .aux file, so that BIBTEX, when it processes a given .aux file, will add the corresponding bibliographical data in the .bbl file but not in the others.
3. Ensure that the resulting .bbl files are read back into LATEX at the right place.
4. Handle the problem of escaping citations due to their placement in sectioning or \caption commands. A citation in such a place would later appear in the table of contents or list of figures, and there (in a different context) LATEX would have problems in resolving it.

The packages chapterbib, bibunits, bibtopic, and multibib, which are described in this section, solve the above problems in different ways. They all have their own advantages and disadvantages. A short comparison of these packages appears in Table 12.2, where blue entries indicate features (or missing features) that may force a selection when one is looking for a solution for bibliographies per unit or with bibliographies per topic, or a combination of both.

12.6.1 chapterbib—Bibliographies per included file

The chapterbib package (developed by Donald Arseneau based on original work by Niel Kempson) allows multiple bibliographies in a LaTeX document, including the same cited items occurring in more than one bibliography.

It solves the problem of producing several `.aux` files for BibTeX, by relying on the `\include` mechanism of LaTeX; you can have one bibliography per `\include`d file. This package can be used, for example, to produce a document with bibliographies per chapter (hence the name), where each chapter is stored in a separate file that is included with the `\include` command. This approach has the following restrictions:

- Each `\include` file needs to have its own `\bibliography` command. The database files that are listed in the argument can, of course, be different in each file. What is not so obvious is that each file must also contain a `\bibliographystyle` command, though for reasons of uniformity *preferably with the same style argument* (Example 12-6-1 on the next page shows that different styles can be applied).
- An `\include` file not containing a `\bibliography` command cannot contain citation commands, as they would not get resolved.
- Citation commands outside of `\include` files (with the exception of those appearing in the table of contents; see below) will not be resolved, unless you include a `thebibliography` environment on that level. Without special precautions, this environment has to be entered manually. If you use BibTeX on the document's `.aux` file you will encounter errors, because BibTeX sees multiple `\bibdata` and `\bibstyle` commands (when processing the included `.aux` files). In addition, you will get *all* citations from *all* `\include` files added, and that is perhaps not desirable. If you do want a cohesive bibliography for the whole document, there is a `rootbib` option to help with this task. However, it requires adding and removing the option at different stages in the process; see the package documentation for details.
- Units containing a local bibliography will always start a new page (because of the `\include` command). For cases where this is not appropriate, chapterbib offers some support through a `\cbinput` command and `cbunit` environment; see the package documentation for details. Unless you need the `gather` option, it might be better to use the bibunits package in such situations.

By default, the `thebibliography` environment generates a numberless heading corresponding to the highest sectioning level available in the document class (e.g., `\chapter*` with the book class). However, if bibliographies are to be generated for individual parts of the document this may not be the right level. In that case you can use the option `sectionbib`[1] to enforce `\section*` headings for the bibliographies.

[1]If both chapterbib and natbib are used, use the `sectionbib` option of natbib instead!

In the following example, we present the \include files article-1.tex and article-2.tex in filecontents environments, which allows us to process this example automatically for the book. In real life these would be different files on your computer file system. We also use \stepcounter to change the chapter counter rather than using \chapter to avoid getting huge chapter headings in the example. Note that both included files refer to a publication with the key Knuth-CT-a. These are actually treated as different keys in the sense that one refers to the publication from article-1.bbl and the other refers to that from article-2.bbl.

... see [Knu86] ...

Bibliography

[Knu86] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

... see [2] and [1] ...

Bibliography

[1] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht*. München, 27. edition, 2002.

[2] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
\begin{filecontents}{article-1.tex}
 \stepcounter{chapter}
 \ldots\ see \cite{Knuth-CT-a} \ldots
 \bibliographystyle{alpha}
 \bibliography{tex}
\end{filecontents}
\begin{filecontents}{article-2.tex}
 \stepcounter{chapter}
 \ldots see \cite{Knuth-CT-a}
  and \cite{bschur} \ldots
 \bibliographystyle{plain}
 \bibliography{tex,jura}
\end{filecontents}

\usepackage[sectionbib]{chapterbib}
\include{article-1}
\include{article-2}
```

12-6-1

If you wish to group all the bibliographies together (for example, at the end of the document), use the option gather and place a \bibliography command at the point where the combined bibliography should appear. The argument to that command can be left empty as it is not used to communicate with BIBTEX.

Instead of gather, you may want to use the option duplicate. It will produce "chapter bibliographies", plus the combined listing. Both options work only in document classes that have a \chapter command. The headings generated by either option can be customized by redefining the command \StartFinalBibs, which is executed at the point where the top-level \bibliography command is encountered. In the following example it generates an unnumbered \chapter heading, sets up the running head via \chaptermark, and then redefines \bibname, which provides the text used in the heading for each sub-bibliography. As you can see \thechapter is used to number the sub-bibliographies, so this mechanism works only if all chapters have bibliographies; otherwise, the numbering will be wrong.

If you do not place the combined bibliography at the end of the document, make sure that \bibname is properly reset afterwards. Otherwise, any subsequent bibliography in an \include file will inherit the modified definition.

If the highest heading unit in your document is `\section`, the redefinition of `\StartFinalBibs` can be done in a similar way. You then have to use `\refname` instead of `\bibname`, since that is the command used in classes derived from the article document class.

References by article

Article 1

[Knu86] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

Article 2

[1] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht*. München, 27. edition, 2002.

12-6-2 [2] Donald E. Knuth. *The TEXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
% included files as in
% previous example
\usepackage
    [gather,sectionbib]
    {chapterbib}
\renewcommand\StartFinalBibs
  {\chapter*
    {References by article}%
   \chaptermark
    {References by article}%
   \renewcommand\bibname
    {Article~\thechapter}}
\include{article-1}
\include{article-2}
\bibliography{}
```

If citations are placed into sectioning or `\caption` commands they will appear eventually in some table of contents list (i.e., at the top level). Nevertheless, chapterbib will properly resolve them, by inserting extra code into `.toc`, `.lof`, and `.lot` files so that a `\cite` command is able to determine to which local bibliography it belongs. If you have additional table of contents lists set up, as explained in Section 2.3.4, you have to be careful to avoid citations that may end up in these new contents lists, as chapterbib is unaware of them.

Some BIBTEX styles unfortunately use `\newcommand` declarations instead of `\providecommand` in the generated `.bbl` files, which makes such files unsuitable for repeated loading. If you get "Command ⟨*name*⟩ already defined" errors for this reason, surround the `\bibliography` commands and their arguments in braces. For example, write

Command already defined error

```
{\bibliography{tex,jura}}
```

The chapterbib package is compatible with most other packages, including the citation packages discussed earlier in this chapter. If you plan to use it together with babel, load the chapterbib package first.

12.6.2 bibunits—Bibliographies for arbitrary units

The bibunits package developed by Thorsten Hansen (from original work by José Alberto Fernández) generates separate bibliographies for different units (parts) of the text (chapters, sections, or `bibunit` environments). The package will separate the citations of each unit of text into a separate file to be processed by BIBTEX. A global bibliography can also appear in the document, and citations can be placed in both at the same time.

One way to denote the units that should have a separate bibliography is by enclosing them in a `bibunit` environment.

`\begin{bibunit}[`*style*`] ... \putbib[`*file-list*`] ... \end{bibunit}`

The optional parameter *style* specifies a style for the bibliography different from a default that may have been set up (see below). Instead of `\bibliography` you use a `\putbib` command to place the bibliography. It can appear anywhere within the unit as proven by the example. The optional argument *file-list* specifies a comma-separated list of BIBTEX database files; again a default can be set up. A default BIBTEX style can be set with `\defaultbibliographystyle`; without it, `plain` is used as the default. Similarly, `\defaultbibliography` can be used to define a default list of BIBTEX databases. In its absence `\jobname.bib` is tried. To be effective the default declarations have to appear after `\begin{document}`.

Setting up defaults

1 First one

[1] was used to produce [2].

References

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Another one

References

[1] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht*. München, 29. edition, 2003.

As described by [1] ...

```
\usepackage{bibunits}
\defaultbibliographystyle{plain}
\section{First one}
\begin{bibunit}[plain]
 \cite{GNUMake} was used to
 produce \cite{Knuth:TB10-1-31}.
 \putbib[tex]
\end{bibunit}
\section{Another one}
\begin{bibunit}[plain]
 \putbib[jura]
 As described by \cite{aschur}
 \ldots
\end{bibunit}
```

12-6-3

For each unit bibunits writes the `\citation` commands (used to communicate with BIBTEX) into the file `bu`⟨*num*⟩`.aux`, where ⟨*num*⟩ is an integer starting with 1. Thus, to generate the necessary bibliographies, you have to run BIBTEX on the files `bu1`, `bu2`, and so forth. As a consequence, with the default settings you cannot process more than one document that uses bibunits in the same directory, as the the auxiliary files would be overwritten.[1]

After generating the bibliographies you have to rerun LATEX at least twice to resolve the new cross-references. Be aware that older versions of the package do *not* warn you about the need for a further rerun.

A global bibliography, in addition to the bibliographies for the individual units, can be generated by using `\bibliography` and `\bibliographystyle` as usual. Outside of a `bibunit` environment, the standard commands should be used

[1] If necessary, you can direct the package to use different names; see the package documentation.

to generate a citation for the global bibliography. Inside `bibunit`, use `\cite*` and `\nocite*` instead of `\cite` and `\nocite` to generate a citation for both the local and the global bibliography. There are, however, a number of restrictions. If the natbib package is also loaded, then `\cite*` has the meaning defined by natbib and cannot be used for generating a global citation (use `\nocite` outside the unit in that case). In addition, refrain from using numerical citation labels, since they are likely to produce ambiguous labels in the global bibliography, as shown in the next example. A better choice would be a BibTeX style such as `alpha`.

1 First one

[1] was used to produce [2].

References

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Another one

As described by [1] ...

References

[1] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

Global References

[1] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

12-6-4

```
\usepackage{bibunits}
\section{First one}
\begin{bibunit}[plain]
 \cite{GNUMake} was used to
 produce \cite*{Knuth:TB10-1-31}.
 \putbib[tex]
\end{bibunit}
\section{Another one}
\begin{bibunit}[plain]
 As described by
 \cite*{Knuth:TB10-1-31}
 \ldots    \putbib[tex]
\end{bibunit}
\renewcommand\refname
      {Global References}
\bibliographystyle{plain}
\bibliography{tex}
```

Rather than using `\cite*` everywhere in your document, you can specify the package option `globalcitecopy`. All local citations are then automatically copied to the global bibliography as well.

Instead of specifying the bibliography units with `bibunit` environments explicitly, you can specify the sectioning unit for which bibliography units should be generated automatically.

`\bibliographyunit[`*unit*`]`

This command specifies for which document unit references must be generated, such as *unit*=`\chapter` (for each chapter) or *unit*=`\section` (for each section). If the optional argument is not given, the command `\bibliographyunit` deactivates further bibliography units. When `\bibliographyunit` is active, the `\bibliographystyle` and `\bibliography` commands specify the BibTeX files and the style to be used by default for a global bibliography, as well as in the local units. If you wish to specify information for local bibliographies only, use `\bibliography*` and `\bibliographystyle*` instead. These declarations *cannot* be used in the preamble but must be placed after `\begin{document}`.

There is, however, a catch with the approach: the normal definition of the `thebibliography` environment, which surrounds the reference lists, generates a heading of the highest sectioning level. Hence, if you use `\chapter` units in a report, the heading generated by that environment will prematurely end the unit and consequently you will end up with undefined references, as shown in the example (using `\section` units in an article class).

Getting unresolved references

1 First one

[?] was used to produce [?].

References

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Another one

As described by [?] ...

References

[1] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht.*

```
\usepackage{bibunits}
\bibliographyunit[\section]
\bibliographystyle*{plain}
\bibliography*{tex,jura}
\section{First one}
  \cite{GNUMake} was
  used to produce
  \cite{Knuth:TB10-1-31}.
  \putbib
\section{Another one}
  As described by
  \cite{aschur} \ldots
  \putbib
```

12-6-5

To resolve this problem, you can provide your own definition for the `thebibliography` environment, so that it uses a different sectioning level than the one specified on the `\bibliographyunit` declaration. Alternatively, you can use the option `sectionbib` (use `\section*` as a heading in `thebibliography`) or `subsectionbib` (use `\subsection*`) to change the `thebibliography` environment for you.

1 First one

[1] was used to produce [2].

References

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Another one

As described by [1] ...

References

[1] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht.* München, 29. edition, 2003.

```
\usepackage[subsectionbib]
            {bibunits}
\bibliographyunit[\section]
\bibliographystyle*{plain}
\bibliography*{tex,jura}
\section{First one}
  \cite{GNUMake} was
  used to produce
  \cite{Knuth:TB10-1-31}.
  \putbib
\section{Another one}
  As described by
  \cite{aschur} \ldots
  \putbib
```

12-6-6

Note that the unit specified on the `\bibliographyunit` command has to be different from the one referred to in the option. In the above example the unit was `\section`, so we used the `subsectionbib` option.

To resolve the problem of escaping citations (see page 746), the package offers the option `labelstoglobalaux`. However, this has the side effects that such citations will appear in the global bibliography and that numerical reference schemes are likely to produce incorrect labels; see the package documentation for details.

12.6.3 bibtopic—Combining references by topic

In contrast to chapterbib and bibunits, which collect citations for individual units of a document, the package bibtopic written by Stefan Ulrich (based on earlier work by Pierre Basso) combines reference listings by topic. You can, for example, provide a primary reference listing separate from a reference list for further reading, or put all references to books separate from those to articles.

Within the document all citations are produced with `\cite`, `\nocite`, or variants thereof (if natbib or similar packages are also loaded). Thus, separation into topics is handled at a later stage. To produce separate bibliographies by topic you have to group the bibliographical entries that belong to one topic in a separate BIBTEX database file (e.g., one for primary sources and one for secondary literature). The bibliographies are then generated by using several `btSect` environments. Ways to generate separate database files are described in Chapter 13. You can, for example, use the program bibtool to extract reference entries according to some criteria from larger BIBTEX database collections.

`\begin{btSect}`[*style*]{*file-list*}

The `btSect` environment generates a bibliography for all citations from the whole document that have entries in the BIBTEX database files listed in the comma-separated *file-list* argument. If the optional *style* argument is present, it specifies the BIBTEX style to use for the current bibliography. Otherwise, the style specified by a previous `\bibliographystyle` declaration is used. If no such declaration was given, the BIBTEX style `plain` is used as a default.

Unless the package was loaded with the option `printheadings` the environment produces no heading. Normally, you have to provide your own heading using `\section*` or a similar command.

`\btPrintCited` `\btPrintNotCited` `\btPrintAll`

Within a `btSect` environment one of the above commands can be used to define which bibliographical entries are included among those from the specified *file-list* databases. The `\btPrintCited` command prints all references from *file-list* that have been somewhere cited in the document, `\btPrintNotCited` prints those that have not been cited, and `\btPrintAll` prints all entries in the BIBTEX database files.

The following example shows the basic concepts using two topics: "TEX related" and "Juridical" literature. The first bibliography uses the default `plain` style; for the second bibliography we explicitly specified the BIBTEX style `abbrv` (this is meant as an illustration—mixing styles is usually a bad idea). As you can see, if you specify numerical BIBTeX styles, bibtopic automatically uses consecutive numbers throughout all bibliographies, to ensure that the references in the document are unique.

We saw the citations [3], [2], and [1].

Juridical literature

[1] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht.* München, 27. edition, 2002.

[2] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht.* München, 29. edition, 2003.

TeX literature

[3] D. E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
\usepackage{bibtopic}
We saw the citations \cite{Knuth-CT-a},
\cite{aschur}, and \cite{bschur}.
\begin{btSect}{jura}
  \section*{Juridical literature}
  \btPrintCited
\end{btSect}
\begin{btSect}[abbrv]{tex}
  \section*{\TeX{} literature}
  \btPrintCited
\end{btSect}
```

12-6-7

For every btSect environment, the bibtopic package generates a separate .aux file that by default is constructed from the base name of the source document (\jobname) and a sequence number. You can change this naming scheme by redefining \thebtauxfile using the counter btauxfile to automatically obtain a sequence number. For the book examples we used the following redefinition:

```
\renewcommand\thebtauxfile{\jobname+\arabic{btauxfile}}
```

Bibliographic topics per logical unit

The bibtopic package is incompatible with chapterbib and bibunits. However, it provides the environment btUnit to confine the citations to logical units. Within such units the btSect environment can be used in the normal way, allowing for topic bibliographies by chapter or other unit. In that case *all* citations have to appear within such units (escaping citations, discussed on page 746, are not handled so you have to ensure that they do not happen). By default, numerical styles restart their numbering per unit (e.g., per article in a proceedings issue). If you want continuous numbering use the option unitcntnoreset.

Problem with nonsorting BIBTeX styles

While bibtopic works with most BIBTeX styles, there are some exceptions. The most important one is that it does not work as expected with "unsorted" styles (e.g., unsrt). If such a style is used, then the order in the bibliography is determined by the order in the BIBTeX database file and *not* by the order of citation in the document. If the latter order is required, you should use the multibib package described in the next section.

The bibtopic package is compatible with most other packages that provide extensions to the citation mechanism, including cite, natbib, and jurabib. There are some restrictions with respect to the production of the bibliography lists. For example, hooks to influence the layout as provided by natbib or jurabib may not be functional. Details are given in the package documentation.

We saw the citations Knuth: The TeXbook and Brox/Walker: Allgemeines Schuldrecht.

TeX literature

Knuth, Donald E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

Juridical literature

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29th edition. München, 2003

12-6-8

```
\usepackage{bibtopic,jurabib}
\bibliographystyle{jurabib}
We saw the citations \cite{Knuth-CT-a}
and \cite{aschur}.
\begin{btSect}{tex}
  \section*{\TeX{} literature}
  \btPrintCited
\end{btSect}
\begin{btSect}{jura}
  \section*{Juridical literature}
  \btPrintCited
\end{btSect}
```

12.6.4 multibib—Separate global bibliographies

Like bibtopic, the multibib package written by Thorsten Hansen provides separate global bibliographies. While the former package separates the bibliographies by using separate BibTeX database files, multibib works by providing separate citation commands to distinguish citations in different bibliographies.

There are advantages and disadvantages with either method. With multibib, different types of citations are clearly marked already in the source document. As a consequence, however, moving a citation from one bibliography to a different one in a consistent manner requires changes to the document in various places. In contrast, with bibtopic it merely requires moving the corresponding database entry from one file to another. On the other hand, bibtopic often requires tailored .bib files for each new document, while with multibib one can use generally available collections of BibTeX database files.

Recent versions of multibib are compatible with most other packages that provide extensions to the cite mechanisms, including cite, jurabib, and natbib. Moreover, the package provides a general interface which allows to add arbitrary extensions of cite commands to be recognized by multibib.

`\newcites{`*type*`}{`*title*`}`

The `\newcites` declaration defines an additional set of citation commands for a new *type* of citations. The heading for the additional bibliography listing is *title*. Once this declaration is given the four additional commands are available for use. The command `\cite`⟨*type*⟩, like `\cite`, generates a citation within the text and its corresponding reference appears in the bibliography listing for the new *type*. Similarly, `\nocite`⟨*type*⟩ adds a citation to the *type* bibliography without appearing in the text. The corresponding bibliography appears at the point where the `\bibliography⟨type⟩` command is given, and the BibTeX style used for this bibliography is defined with `\bibliographystyle⟨type⟩`. An example is shown below.

A book on graphics in LaTeX is [1]; suggestions on citations can be found in [vL92].

LaTeX references

[1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript.* Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997.

General references

[vL92] Mary-Claire van Leunen. *A handbook for scholars.* Oxford University Press, Walton Street, Oxford OX2 6DP, UK, 92.

```
\usepackage{multibib}
\newcites{latex}
         {\LaTeX{} references}
A book on graphics in \LaTeX{} is
\citelatex{LGC97}; suggestions on
citations can be found in
\cite{vLeunen:92}.

\bibliographystylelatex{plain}
\bibliographylatex{tex}

\renewcommand\refname
              {General references}
\bibliographystyle{alpha}
\bibliography{tex}
```

12-6-9

The `\newcites` declaration can be used several times, thereby creating additional citation types. It is limited only by the number of output files that can be used simultaneously by TeX. The `.aux` file written for communication with BibTeX has the name `⟨type⟩.aux`. For this reason one has to be a bit careful when selecting the *type* in the first argument to `\newcites`, to avoid overwriting other `.aux` files.

For numerical citation styles the references are by default numbered sequentially over all bibliographies to avoid ambiguous references. When using the option `resetlabels`, each bibliography restarts the numbering.

LaTeX offers an interface to include graphics.[1] LaTeX's default citation scheme is number-only.[2]

LaTeX references

[1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript.* Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997.

General references

[2] Mary-Claire van Leunen. *A handbook for scholars.* Oxford University Press, Walton Street, Oxford OX2 6DP, UK, 92.

```
\usepackage[super]{cite}
\usepackage{multibib}
\newcites{latex}{\LaTeX{} references}
\LaTeX{} offers an interface to include
graphics \citelatex{LGC97}. \LaTeX's
default citation scheme is
number-only \cite{vLeunen:92}.

\bibliographystylelatex{plain}
\bibliographylatex{tex}

\renewcommand\refname
              {General references}
\bibliographystyle{plain}
\bibliography{tex}
```

12-6-10

CHAPTER 13

Bibliography Generation

While a table of contents (see Section 2.3) and an index (discussed in Chapter 11) make it easier to navigate through a book, the presence of bibliographic references should allow you to verify the used sources and to probe further subjects you consider interesting. To make this possible, the references should be precise and lead to the relevant work with a minimum of effort.

There exist many ways for formatting bibliographies, and different fields of scholarly activities have developed very precise rules in this area. An interesting overview of Anglo-Saxon practices can be found in the chapter on bibliographies in *The Chicago Manual of Style* [38]. Normally, authors must follow the rules laid out by their publisher. Therefore, one of the more important tasks when submitting a book or an article for publication is to generate the bibliographic reference list according to those rules.

Traditional ways of composing such lists by hand, without the systematic help of computers, are plagued with the following problems:

- Citations, particularly in a document with contributions from many authors, are hard to make consistent. Difficulties arise, such as variations in the use of full forenames versus abbreviations (with or without periods); italicization or quoting of titles; spelling "ed.", "Ed.", or "Editor"; and the various forms of journal volume number.
- A bibliography laid out in one style (e.g., alphabetic by author and year) is extremely hard to convert to another (e.g., numeric citation order) if requested by a publisher.
- It is difficult to maintain one large database of bibliographic references that can be reused in different documents.

In Chapter 12 we were mainly concerned with the citation of sources within the text. In the present chapter we concentrate on the formatting of reference lists and bibliographies, and we discuss possibilities for managing collections of citations in databases. The chapter is heavily based on the BibTeX program, written by Oren Patashnik, which integrates well with LaTeX.

We start by introducing the program and variants of it, touching on recent developments geared toward creating a successor. This is followed by a detailed introduction to the BibTeX database format, which collects information on how to specify bibliographical data in a suitable form to be processed by BibTeX. Instead of collecting your own bibliographical data, there is also the possibility of drawing information from various on-line sources that offer such data in BibTeX format. Some of them are introduced in Section 13.3.

Having collected data for BibTeX databases, the next natural step is to look for tools that help in managing such databases. Section 13.4 offers tools of various flavors for this task, ranging from command-line utilities to GUI-based programs for various platforms.

Once everything is under control, we return in Section 13.5 to the task of typesetting and look at how different BibTeX styles can be used to produce different bibliography layouts from the same input. As there may not be a suitable style for a particular set of layout requirements available, Section 13.5.2 discusses how to generate customized styles using the custom-bib package without the need for any BibTeX style programming.

For those readers who really want to (or have to) dig into the mysteries of BibTeX style programming, the final section gives more details about the format of such style files, including a short overview of the commands and intrinsic functions available. The global structure of the generic style documentation file `btxbst.doc` is explained, and it is shown how to adapt an existing style file to the needs of a particular house style or foreign language.

13.1 The BibTeX program and some variants

The BibTeX program was designed by Oren Patashnik to provide a flexible solution to the problem of automatically generating bibliography lists conforming to different layout styles. It automatically detects the citation requests in a LaTeX document (by scanning its `.aux` file or files), selects the needed bibliographical information from one or more specified databases, and formats it according to a specified layout style. Its output is a file containing the bibliography listing as LaTeX code that will be automatically loaded and used by LaTeX on the following run. Section 12.1.3 on page 687 discussed the interface between the two programs in some detail.

At the time of this book's writing BibTeX was available as version `0.99c`, but if you look into the first edition of this book (a decade back), you will find that it also talks about version `0.99c`. The version `0.99a` probably dates back to 1986. In other words, the program has been kept stable for a very long period of time. As a consequence, the BibTeX database format is very well established in the LaTeX world, with many people having numerous citation entries collected over the years. Thus, it comes as no surprise that all development that happened in the last decade is based on that format as a standard.

In this section we briefly survey a number of developments in this arena. Some new projects have surfaced especially in recent years, but there are also some projects that date back a few years.

13.1.1 bibtex8—An 8-bit reimplementation of BibTeX

Due to its age and origins BibTeX is 7-bit, ASCII based. Although it is able to handle foreign characters, its functionality in this respect is rather limited. The BibTeX8 program written by Niel Kempson and Alejandro Aguilar-Sierra is an 8-bit reimplementation of BibTeX with the ability to specify sorting order information. This allows you to store your BibTeX database entries in your favorite 8-bit code page, and to use the inputenc package in your LaTeX document (see Sections 7.5.2 and 7.11.3). Sorting order information related to a specific encoding can be specified on the command line—for example,

```
bibtex8 -c 88591lat tlc2
```

on the author's machine. The sorting order is stored in files with the extension `.csf` (e.g., in the above example in the file `88591lat.csf`). The distribution comes with a number of such files for the most popular encodings. The format is well documented so that it should be possible to provide your own `.csf` file if necessary. Related command-line options are `-7` and `-8` to force 7-bit or 8-bit processing, respectively, without a special sorting order.

The BibTeX8 program offers a second set of command-line options that allows you to enlarge its internal tables. In 1995, when the first release of the program was written, standard BibTeX had only small, hard-wired internal tables, making it impossible to typeset, say, a bibliography listing with several hundred citations. These days most installations use higher compile defaults (e.g., 5000 citations) so that the flexibility of BibTeX8 in this respect is seldom needed. But in case a particular job hits one of the limits and emits a message like "`Sorry—you've exceeded BibTeX's...`" you can use BibTeX8 with a suitable command-line setting to get around the problem. You can find out about the possible options by calling the program without any input or with the option `-h` or `--help`.

13.1.2 Recent developments

Besides BibTeX and BibTeX8, both of which have been available for a long time, there have been some more recent developments that target bibliography generation. In this section we briefly introduce three projects that might be of interest to the reader. It is quite possible that one or the other project merge together in the future, so this list should be viewed as a snapshot of the situation in 2003 and as proof that there is a renewed interest in further development.

bibulus—Bibliographies with XML and perl

The program bibulus by Thomas Widmann is a BIBTEX replacement written in perl.[1] It does not use BIBTEX's database file format but rather works with bibliographical entries stored in XML format and provides its own document type definition (`bibulus.dtd`). This way bibliographical entries can be manipulated and processed with any application that understands XML. To enable the reuse of existing `.bib` files, the program provides a tool to convert your BIBTEX databases to XML format.

The bibulus program uses Unicode internally and thus is truly multilingual; at the same time it is able to read and write output in other encodings. The textual strings generated by the program have been translated into a large number of languages. The current implementation of bibulus provides support for more than a dozen languages.

From the program's point of view LATEX is only one of the different possible target output formats. Alternatives range from plain text output, to HTML, to input formats for other programs dealing with citations.

Like the other two programs described below, bibulus is work in progress. It is available from `http://www.nongnu.org/bibulus`, where you will also find further information on the project.

BIBTEX++—A BIBTEX successor in Java

The BIBTEX++ project is a Java-based implementation of a citation manager written by Emmanuel Donin de Rosière in the course of a master thesis [146] supervised by Ronan Keryell. Being intended to serve as a BIBTEX successor, it can, of course, be used in the LATEX world, but it also accepts other bibliography formats and different style languages and can produce output for several typesetting systems. The program is integrated in a web-based environment, so it can retrieve lacking information from various Internet sources. BIBTEX++ uses a plug-in concept that allows you to dynamically extend its functionalities, perhaps to support special formatting conventions or to generate output for other formatters.

Existing BIBTEX style files can be converted to a BIBTEX++ style using a translation program that was developed as part of the project. The result can be further customized by using the BIBTEX++ concepts, thus easing the initial development of a new style.

The project's home is at `http://bibtex.enstb.org`, where you will find a CVS repository as well as compiled binaries and further information.

[1] For installation and use it needs a recent perl implementation (5.8+).

MlBIBTEX—A multilingual successor of BIBTEX

The program MlBIBTEX, developed by Jean-Michel Hufflen, is a reimplementation and extension of BIBTEX with particular focus on multilingual features. A first release became available in 2001. However, the author found that the approach taken back then was not really suitable for the typographical conventions used in some languages. At that stage of the project he developed a questionnaire to obtain more insight into the problems and conventions with bibliographic data in different European countries. In response, a new implementation was started; its first results were presented at various conferences in 2003.

The current release (v1.3) implements a style language named `nbst`, for specifying layout and formatting directives. This language is close, but not identical, to XSLT, the language for manipulating and processing XML documents.

The project's home is at `http://lifc.univ-fcomte.fr/~hufflen/texts/mlbibtex/mlbibtex/mlbibtex.html`, where further information can be found.

13.2 The BIBTEX database format

A BIBTEX database is a plain text (ASCII) file that contains bibliographical entries internally structured as keyword/value pairs. A typical database file was shown in Figure 12.2 on page 690. In this section we study the allowed syntax of its entries in some detail; see also [135].

Each entry in a BIBTEX database consists of three main parts: a *type* specifier, followed by a *key*, and finally the *data* for the entry itself. The *type* describes the general nature of the entry (e.g., whether it is an article, book, or some other publication). The *key* is used in the interface to LATEX; it is the string that you have to place in the argument of a `\cite` command when referencing that particular entry. The *data* part consists of a series of *field entries* (depending on the *type*), which can have one of two forms as seen in the following generic format and example:

```
@type_specifier{key_identifier,      @book{lamport86,
  field_name_1 = "field_text_1",       author = "Leslie Lamport",
  field_name_2 = {field_text_2},       title  = "{\LaTeX{}} A Document
       . . .                                      Preparation system",
  field_name_n = {field_text_n}        publisher = {Addison-Wesley},
}                                      year   = 1986               }
```

The comma is the field separator. Spaces surrounding the equals sign or the comma are ignored. Inside the text part of a field (enclosed in a pair of double quotes or a pair of braces) you can have any string of characters, but braces must be matched. The quotes or braces can be omitted for text consisting entirely of numbers (like the `year` field in the example above). Note that LATEX's comment character % is not a comment character inside .bib database files. Instead, anything outside an entry is considered a comment as long as it does not contain an @ sign (which would be misinterpreted as the start of a new entry).

BibTeX *ignores* the case of the letters for the entry type, key, and field names. You must, however, be careful with the key. LaTeX honors the case of the keys specified as the argument of a \cite command, so the key for a given bibliographic entry must match the one specified in the LaTeX file (see Section 12.2.1).

13.2.1 Entry types and fields

As discussed above, you must describe each bibliographic entry as belonging to a certain class, with the information itself tagged by certain fields.

The first thing you have to decide is what type of entry you are dealing with. Although no fixed classification scheme can be complete, with a little creativity you can make BibTeX cope reasonably well with even the more bizarre types of publications. For nonstandard types, it is probably wise not to attach too much importance to BibTeX's warning messages (see below).

Most BibTeX styles have at least the 13 standard entry types, which are shown in Table 13.1 on the facing page. These different types of publications demand different kinds of information; a reference to a journal article might include the volume and number of the journal, which is usually not meaningful for a book. Therefore, different database types have different fields. In fact, for each type of entry, the fields are divided into three classes:

Required Omission of the field will produce a warning message and, possibly, a badly formatted bibliography entry. If the required information is not meaningful, you are using the wrong entry type. If the required information is meaningful but, say, already included in some other field, simply ignore the warning.

Optional The field's information will be used if present, but you can omit it without causing formatting problems. Include the optional field if it can help the reader.

Ignored The field is ignored. BibTeX ignores any field that is not required or optional, so you can include any fields in a .bib file entry. It is a good idea to put all relevant information about a reference in its .bib file entry, even information that may never appear in the bibliography. For example, the abstract of a paper can be entered into an abstract field in its .bib file entry. The .bib file is probably as good a place as any for the abstract, and there exist bibliography styles for printing selected abstracts (see the abstract bibliography style mentioned in Table 13.4 on page 791).

Table 13.1 on the facing page describes the standard entry types, along with their required and optional fields, as used by the standard bibliography styles.

article	An article from a journal or magazine. *Required*: author, title, journal, year. *Optional*: volume, number, pages, month, note.
book	A book with an explicit publisher. *Required*: author or editor, title, publisher, year. *Optional*: volume or number, series, address, edition, month, note.
booklet	A work that is printed and bound, but without a named publisher or sponsoring institution. *Required*: title. *Optional*: author, howpublished, address, month, year, note.
inbook	A part of a book, e.g., a chapter, section, or whatever and/or a range of pages. *Required*: author or editor, title, chapter and/or pages, publisher, year. *Optional*: volume or number, series, type, address, edition, month, note.
incollection	A part of a book having its own title. *Required*: author, title, booktitle, publisher, year. *Optional*: editor, volume or number, series, type, chapter, pages, address, edition, month, note.
inproceedings	An article in a conference proceedings. *Required*: author, title, booktitle, year. *Optional*: editor, volume or number, series, pages, address, month, organization, publisher, note.
manual	Technical documentation. *Required*: title. *Optional*: author, organization, address, edition, month, year, note.
mastersthesis	A master's thesis. *Required*: author, title, school, year. *Optional*: type, address, month, note.
misc	Use this type when nothing else fits. A warning will be issued if all optional fields are empty (i.e., the entire entry is empty or has only ignored fields). *Required*: none. *Optional*: author, title, howpublished, month, year, note.
phdthesis	A Ph.D. thesis. *Required*: author, title, school, year. *Optional*: type, address, month, note.
proceedings	Conference proceedings. *Required*: title, year. *Optional*: editor, volume or number, series, address, publisher, note, month, organization.
techreport	A report published by a school or other institution, usually numbered within a series. *Required*: author, title, institution, year. *Optional*: type, number, address, month, note.
unpublished	A document having an author and title, but not formally published. *Required*: author, title, note. *Optional*: month, year.

Table 13.1: BibTeX's entry types as defined in most styles

The fields within each class (required or optional) are listed in the typical order of occurrence in the output. A few entry types, however, may perturb the alphabetic ordering slightly, depending on which fields are missing. The meaning of the individual fields is explained in Table 13.2 on the next page. Nonstandard bibliography styles may ignore some optional fields or use additional ones like `isbn` when creating the reference (see also the examples starting on page 793). Remember that, when used in a `.bib` file, the entry-type name is preceded by an `@` character.

Sorting of entries

Most BIBTEX style files sort the bibliographical entries. This is done by internally generating a sort key from the author's/editor's name, the date of the publication, the title, and other information. Entries with identical sort keys will appear in citation order.

The author information is usually the `author` field, but some styles use the `editor` or `organization` field. In addition to the fields listed in Table 13.1, each entry type has an optional `key` field, used in some styles for alphabetizing, for cross-referencing, or for forming a `\bibitem` label. You should therefore include a `key` field for any entry whose author information is missing. Depending on the style the `key` field can also be used to overwrite the automatically generated internal key for sorting.[1] A situation where a `key` field is useful is the following:

```
organization = "The Association for Computing Machinery",
key = "ACM"
```

Without the `key` field, the `alpha` style would construct a label from the first three letters of the information in the `organization` field. Although the style file will strip off the article "The", you would still get a rather uninformative label like "[Ass86]". The `key` field above yields a more acceptable "[ACM86]".

We now turn our attention to the fields recognized by the standard bibliography styles. These "standard" fields are shown in Table 13.2 on the facing page. Other fields, like `abstract`, can be required if you use one of the extended nonstandard styles shown in Table 13.4 on page 791. As nonrecognized fields are ignored by the BIBTEX styles, you can use this feature to include "comments" inside an entry: it is enough to put the information to be ignored inside braces following a field name (and = sign) that is not recognized by the BIBTEX style.

As with the names of the entry types in Table 13.1 on the preceding page, the names of the fields should be interpreted in their widest sense to make them applicable in a maximum number of situations. And you should never forget that a judicious use of the `note` field can solve even the more complicated cases.

13.2.2 The text part of a field explained

The text part of a field in a BIBTEX entry is enclosed in a pair of double quotes or curly braces. Part of the text itself is said to be *enclosed in braces* if it lies inside a matching pair of braces other than the ones enclosing the entire entry.

[1]Some BIBTEX styles (e.g., `jurabib`) use the `sortkey` field instead.

address	Usually the address of the publisher or other institution. For major publishing houses, just give the city. For small publishers, specifying the complete address might help the reader.
annote	An annotation. Not used by the standard bibliography styles, but used by others that produce an annotated bibliography (e.g., annote). The field starts a new sentence and hence the first word should be capitalized.
author	The name(s) of the author(s), in BibTeX name format (Section 13.2.2).
booktitle	Title of a book, part of which is being cited (Section 13.2.2). For book entries use the title field.
chapter	A chapter (or section or whatever) number.
crossref	The database key of the entry being cross-referenced (Section 13.2.5).
edition	The edition of a book (e.g., "Second"). This should be an ordinal, and should have the first letter capitalized, as shown above; the standard styles convert to lowercase when necessary.
editor	Name(s) of editor(s), in BibTeX name format. If there is also an author field, then the editor field gives the editor of the book or collection in which the reference appears.
howpublished	How something strange has been published.
institution	Institution sponsoring a technical report.
journal	Journal name. Abbreviations are provided for many journals (Section 13.2.3).
key	Used for alphabetizing, cross-referencing, and creating a label when the author and editor information is missing. This field should not be confused with the key that appears in the \cite command and at the beginning of the database entry.
month	The month in which the work was published or, for an unpublished work, in which it was written. For reasons of consistency the standard three-letter abbreviations (jan, feb, mar, etc.) should be used (Section 13.2.3).
note	Any additional information that can help the reader.
number	The number of a journal, magazine, technical report, or work in a series. An issue of a journal or magazine is usually identified by its volume and number; a technical report normally has a number; and sometimes books in a named series carry numbers.
organization	The organization that sponsors a conference or that publishes a manual.
pages	One or more page numbers or range of numbers (e.g., 42-111 or 7,41,73-97 or 43+, where the '+' indicates pages that do not form a simple range).
publisher	The publisher's name.
school	The name of the school where the thesis was written.
series	The name of a series or set of books. When citing an entire book, the title field gives its title and an optional series field gives the name of a series or multivolume set in which the book is published.
title	The work's title, typed as explained in Section 13.2.2.
type	The type of a technical report (e.g., "Research Note"). This name is used instead of the default "Technical Report". For the entry type phdthesis you could use the term "Ph.D. dissertation" by specifying: type = "{Ph.D.} dissertation". Similarly, for the inbook and incollection entry types you can get "section 1.2" instead of the default "chapter 1.2" with chapter = "1.2" and type = "Section".
volume	The volume of a journal or multivolume book.
year	The year of publication or, for an unpublished work, the year it was written. Generally, it should consist of four numerals, such as 1984, although the standard styles can handle any year whose last four nonpunctuation characters are numerals, such as "about 1984".

Table 13.2: BibTeX's standard entry fields

The structure of a name

The author and editor fields contain a list of names. The exact format in which these names are typeset is decided by the bibliography style. The entry in the .bib database tells BIBTEX what the name is. You should always type names exactly as they appear in the cited work, even when they have slightly different forms in two works. For example:

```
author = "Donald E. Knuth"            author = "D. E. Knuth"
```

If you are sure that both authors are the same person, then you could list both in the form that the author prefers (say, Donald E. Knuth), but you should always indicate (e.g., in our second case) that the original publication had a different form.

```
author = "D[onald] E. Knuth"
```

BIBTEX alphabetizes this as if the brackets were not there, so that no ambiguity arises as to the identity of the author.

Most names can be entered in the following two equivalent forms:

```
"John Chris Smith"                  "Smith, John Chris"
"Thomas von Neumann"                "von Neumann, Thomas"
```

The second form, with a comma, should always be used for people who have multiple last names that are capitalized. For example,

```
"Lopez Fernandez, Miguel"
```

If you enter "Miguel Lopez Fernandez", BIBTEX will take "Lopez" as the middle name, which is wrong in this case. When the other parts are not capitalized, no such problem occurs (e.g., "Johann von Bergen" or "Pierre de la Porte").

If several words of a name have to be grouped, they should be enclosed in braces. BIBTEX treats everything inside braces as a single name, as shown below.

```
"{Boss and Friends, Inc.} and {Snoozy and Boys, Ltd.}"
```

In this case, Inc. and Ltd. are not mistakenly considered as first names.

In general, BIBTEX names can have four distinct parts, denoted as First, von, Last, and Jr. Each part consists of a list of name tokens, and any list but Last can be empty. Thus, the two entries below are different:

```
"von der Schmidt, Alex"        "{von der Schmidt}, Alex"
```

The first has von, Last, and First parts, while the second has only First and Last parts (von der Schmidt), resulting possibly in a different sorting order.

A "Junior" part can pose a special problem. Most people with "Jr." in their name precede it with a comma, thus entering it as follows:

```
"Smith, Jr., Robert"
```

Certain people do not use the comma, and these cases are handled by considering the "Jr." as part of the last name:

```
"{Lincoln Jr.}, John P."        "John P. {Lincoln Jr.}"
```

Recall that in the case of "Miguel Lopez Fernandez", you should specify

```
"Lopez Fernandez, Miguel"
```

The `First` part of his name has the single token "Miguel"; the `Last` part has two tokens, "Lopez" and "Fernandez"; and the `von` and `Jr` parts are empty.

A complex example is

```
"Johannes Martinus Albertus van de Groene Heide"
```

This name has three tokens in the `First` part, two in the `von` part, and two in the `Last` part. BibTeX knows where one part ends and the other begins because the tokens in the `von` part begin with lowercase letters (`van de` in this example).

In general, `von` tokens have the first letter at brace-level 0 in lowercase. Technically speaking, everything in a "special character" is at brace-level 0 (see page 768), so you can decide how BibTeX treats a token by inserting a dummy special character whose first letter past the TeX control sequence is in the desired case, upper or lower. For example, in

```
Maria {\MakeUppercase{d}e La} Cruz
```

BibTeX will take the uppercase "De La" as the `von` part, since the first character following the control sequence is lowercase. With the `abbrv` style you will get the correct abbreviation M. De La Cruz, instead of the incorrect M. D. L. Cruz if you did not use this trick.

BibTeX handles hyphenated names correctly. For example, an entry like

```
author = "Maria-Victoria Delgrande",
```

with the `abbrv` style, results in "M.-V. Delgrande".

When multiple authors are present, their names should be separated with the word "and", where the "and" must not be enclosed in braces.

```
author = "Frank Mittelbach and Rowley, Chris"
editor = "{Lion and Noble, Ltd.}"
```

There are two authors, Frank Mittelbach and Chris Rowley, but only one editor, since the "and" is enclosed in braces. If the number of authors or editors is too large to be typed *in extenso*, then the list of names can be ended with the string "`and others`", which is converted by the standard styles into the familiar "*et al.*"

To summarize, you can specify names in BIBTEX using three possible forms (the double quotes and curly braces can be used in all cases):

```
"First von Last"           e.g. {Johan van der Winden}
"von Last, First"          e.g. "von der Schmidt, Alexander"
"von Last, Jr, First"      e.g. {de la Porte, Fils, {\'Emile}}
```

The first form can almost always be used. It is, however, not suitable when there is a `Jr` part, or when the `Last` part has multiple tokens and there is no von part.

The format of the title

The bibliography style decides whether a title is capitalized. Usually, titles of books are capitalized, but those for articles are not. A title should always be typed as it appears in the original work. For example:

```
TITLE = "A Manual of Style"
TITLE = "Hyphenation patterns for ancient Greek and Latin"
```

Different languages and styles have their own capitalization rules. If you want to override the decisions of the bibliography style, then you should enclose the parts that should remain unchanged inside braces. Note that this will not be sufficient when the first character after the left brace is a backslash (see below). It is usually best to enclose whole words in braces, because otherwise LATEX may lose kerning or ligatures when typesetting the word. In the following example, the first version is preferable over the second:

```
TITLE = "The Towns and Villages of {Belgium}"
TITLE = "The Towns and Villages of {B}elgium"
```

Accented and special characters

BIBTEX accepts accented characters. If you have an entry with two fields

```
author = "Kurt G{\"o}del",
year = 1931,
```

then the `alpha` bibliography style will yield the label [Göd31], which is probably what you want. As shown in the example above, the entire accented character must be placed in braces; in this case either `{\"o}` or `{\"{o}}` will work. These braces must not themselves be enclosed in braces (other than the ones that might delimit the entire field or the entire entry); also, a backslash must be the very first character inside the braces. Thus, neither `{G{\"{o}}del}` nor `{G\"{o}del}` works here.

This feature handles accented characters and foreign symbols used with LaTeX. It also allows user-defined "accents". For purposes of counting letters in labels, BibTeX considers everything inside the braces to be a single letter. To BibTeX, an accented character is a special case of a "special character", which consists of everything from a left brace at the topmost level, immediately followed by a backslash, up through the matching right brace. For example, the field

```
author = "\OE{le} {\'{E}mile} {Ren\'{e}} van R{\i\j}den"
```

has two special characters: "`{\'{E}mile}`" and "`{\i\j}`".

In general, BibTeX does not process TeX or LaTeX control sequences inside a special character, but it will process other characters. Thus, a style that converts all titles to lowercase transforms

"`The {\TeX BOOK\NOOP} Saga`" into "`The {\TeX book\NOOP} saga`"

The article "The" remains capitalized because it is the first word in the title.

The special character scheme has its uses for handling accented characters (although the introduction of additional braces may upset the generation of ligatures and kerns). It may help to make BibTeX's alphabetizing do what you want, but again with some caveats; see the discussion of the `\SortNoop` command on page 771. Also, since BibTeX counts an entire special character as just one letter, you can force extra characters inside labels.

13.2.3 Abbreviations in BibTeX

BibTeX text fields can be abbreviated. An abbreviation is a string of ASCII characters starting with a letter and not containing a space or any of the following 10 characters:

```
" # % ' ( ) , = { }
```

You can define your own abbreviations with the `@string` command in a `.bib` file, as shown below.

```
@string{AW         = "Addison--Wesley Publishing Company"}
@STRING{cacm       = "Communications of the ACM"}
@String{pub-AW     = {{Ad\-di\-son-Wes\-ley}}}
@String{pub-AW:adr = "Reading, MA, USA"}
@String{TUG        = "\TeX{} Users Group"}
@String{TUG:adr    = {Providence, RI, USA}}
```

Abbreviations can be used in the text part of BibTeX fields, but they should not be enclosed in braces or quotation marks. With the above string definitions, the following two ways of specifying the `journal` field are equivalent:

```
journal = "Communications of the ACM"
journal = cacm
```

The case of the name for an abbreviation is not important, so `CACM` and `cacm` are considered identical, but BibTeX produces a warning if you mix different cases. Also, the `@string` command itself can be spelled as all lowercase, all uppercase, or a mixture of the two cases.

`@string` commands can appear anywhere in the `.bib` file, but an abbreviation must be defined before it is used. It is good practice to group all `@string` commands at the beginning of a `.bib` file, or to place them in a dedicated `.bib` file containing only a list of abbreviations. The `@string` commands defined in the `.bib` file take precedence over definitions in the style files.

You can concatenate several strings (or `@string` definitions) using the concatenation operator `#`. Given the definition

```
@STRING{TUB = {TUGboat }}
```

you can easily construct nearly identical `journal` fields for different entries:

```
@article(tub-98,   journal = TUB # 1998,  ...
@article(tub-99,   journal = TUB # 1999,  ...
@article(tub-00,   journal = TUB # 2000,  ...
```

Most bibliography styles contain a series of predefined abbreviations. As a convention, there should always be three-letter abbreviations for the months: `jan`, `feb`, `mar`, and so forth. In your BibTeX database files you should always use these three-letter abbreviations for the months, rather than spelling them explicitly. This assures consistency inside your bibliography. Information about the day of the month is usually best included in the `month` field. You might, for example, make use of the possibility of concatenation:

```
month = apr # "~1,"
```

Names of popular journals in a given application field are also made available as abbreviations in most styles. To identify them you should consult the documentation associated with the bibliographic style in question. The set of journals listed in Table 13.3 on the facing page should be available in all styles. You can easily define your own set of journal abbreviations by putting them in `@string` commands in their own database file and listing this database file as an argument to LaTeX's `\bibliography` command.

acmcs	ACM Computing Surveys
acta	Acta Informatica
cacm	Communications of the ACM
ibmjrd	IBM Journal of Research and Development
ibmsj	IBM Systems Journal
ieeese	IEEE Transactions on Software Engineering
ieeetc	IEEE Transactions on Computers
ieeetcad	IEEE Transactions on Computer-Aided Design of Integrated Circuits
ipl	Information Processing Letters
jacm	Journal of the ACM
jcss	Journal of Computer and System Sciences
scp	Science of Computer Programming
sicomp	SIAM Journal on Computing
tocs	ACM Transactions on Computer Systems
tods	ACM Transactions on Database Systems
tog	ACM Transactions on Graphics
toms	ACM Transactions on Mathematical Software
toois	ACM Transactions on Office Information Systems
toplas	ACM Transactions on Programming Languages and Systems
tcs	Theoretical Computer Science

Table 13.3: Predefined journal strings in BibTeX styles

13.2.4 The BibTeX preamble

BibTeX offers a `@preamble` command with a syntax similar to that of the `@string` command except that there is no name or equals sign, just the string. For example:

```
@preamble{ "\providecommand\url[1]{\texttt{#1}}"   #
           "\providecommand\SortNoop[1]{}"            }
```

You can see how the different command definitions inside the `@preamble` are concatenated using the # symbol. The standard styles output the argument of the `@preamble` literally to the `.bbl` file, so that the command definitions are available when LaTeX reads the file. If you add LaTeX commands in this way, you must ensure that they are added using `\providecommand` and not `\newcommand`. There are two reasons for this requirement. First, you deprive yourself of the ability to change the definition in the document (e.g., the bibliography might add a simple definition for the command `\url` that you may want to replace by the definition from the url package). Second, sometimes the bibliography is read in several times (e.g., with the chapterbib package), an operation that would fail if `\newcommand` is used.

The other example command used above (`\SortNoop`) was suggested by Oren Patashnik to guide BibTeX's sorting algorithm in difficult cases. This algorithm normally does an acceptable job, but sometimes you might want to override BibTeX's decision by specifying your own sorting key. This trick can be used with foreign languages, which have sorting rules different from those of English, or when you want to order the various volumes of a book in a way given by their original date of publication and independently of their re-edition dates.

Suppose that the first volume of a book was originally published in 1986, with a second edition appearing in 1991, and the second volume was published in 1990. Then you could write

```
@book{ ...  volume=1,  year = "{\SortNoop{86}}1991"  ...
@book{ ...  volume=2,  year = "{\SortNoop{90}}1990"  ...
```

According to the definition of `\SortNoop`, LaTeX throws away its argument and ends up printing only the true year for these fields. For BibTeX `\SortNoop` is an "accent"; thus, it will sort the works according to the numbers 861991 and 901990, placing volume 1 before volume 2, just as you want.

Be aware that the above trick may not function with newer BibTeX styles (for example, those generated with custom-bib) and that some styles have added a `sortkey` field that solves such problems in a far cleaner fashion.

13.2.5 Cross-referencing entries

BibTeX entries can be cross-referenced. Suppose you specify `\cite{Wood:color}` in your document, and you have the following two entries in the database file:

```
@Inbook{Wood:color, author = {Pat Wood}, crossref={Roth:postscript},
        title = {PostScript Color Separation}, pages={201--225}}
@Book{Roth:postscript, editor = {Stephen E. Roth}, title =
      {{Real World PostScript}}, booktitle = {{Real World PostScript}},
      publisher=AW, address=AW:adr, year=1988, ISBN={0-201-06663-7}}
```

The special `crossref` field tells BibTeX that the `Wood:color` entry should inherit missing fields from the entry it cross-references—`Roth:postscript`. BibTeX automatically puts the `Roth:postscript` entry into the reference list if it is cross-referenced by a certain number of entries (default 2) on a `\cite` or `\nocite` command, even if the `Roth:postscript` entry itself is never the argument of a `\cite` or `\nocite` command. Thus, with the default settings, `Roth:postscript` will automatically appear on the reference list if one other entry besides `Wood:color` cross-references it.

The default is compiled into the BibTeX program, but on modern installations[1] it can be changed on the command-line by specifying `--min-crossrefs` together with the desired value:

```
bibtex --min-crossrefs=1 12-5-41
```

For instance, the bibliography for Example 12-5-41 from page 738 was produced with the above setting to ensure that the proceedings entry was typeset as a separate reference even though there was only one cross-reference to it. On the other hand, if you want to avoid a separate entry for the whole proceedings regardless

[1] In BibTeX8 this option is named `-min_crossrefs` or `-M`.

of the number of entries referencing it, set the `--min-crossrefs` option to a suitably large value (e.g., 500).

A cross-referenced entry must occur later in the database files than every entry that cross-references it. Thus, all cross-referenced entries could be put at the end of the database. Cross-referenced entries cannot themselves cross-reference another entry.

You can also use LaTeX's `\cite` command inside the fields of your BibTeX entries. This can be useful if you want to reference some other relevant material inside a note field:

```
note = "See Eijkhout~\cite{Eijkhout:1991} for more details"
```

However, such usage may mean that you need additional LaTeX and BibTeX runs to compile your document properly. This will happen if the citation put into the `.bbl` file by BibTeX refers to a key that was not used in a citation in the main document. Thus, LaTeX will be unable to resolve this reference in the following run and will need an additional BibTeX and two additional LaTeX runs thereafter.

13.3 On-line bibliographies

If you search the Internet you will find a large number of bibliography entries for both primary and secondary publications in free as well as commercial databases. In this section we mention a few free resources on scientific publications that offer bibliographic data in BibTeX and some other formats.

Nelson Beebe maintains nearly 400 BibTeX databases related to scientific journals and particular scientific topics.[1] These range from "Acta Informatica" and "Ada User Journal" to "X Journal" and "X Resource [journal]". All are available as `.bib` source file, `.html`, `.pdf`, and `.ps` listings.

Nelson Beebe's most interesting `.bib` databases, as far as TeX is concerned, are the files `texbook2.bib` and `texbook3.bib` (books about TeX, METAFONT, and friends), `type.bib` (a list of articles and books about typography), `gut.bib` (the contents of the French *Cahiers Gutenberg* journal), `komoedie.bib` (the contents of the German *Die TeXnische Komödie* journal), `texgraph.bib` (sources explaining how to make TeX and graphics work together), `texjourn.bib` (a list of journals accepting TeX as input), `tugboat.bib` (all the articles in *TUGboat*), and `standard.bib` (software standards). The web resources provided by Nelson Beebe also include a series of BibTeX styles and many command-line tools for manipulating bibliography data (discussed in Section 13.4.3).

The Collection of Computer Science Bibliographies by Alf-Christian Achilles, containing more than 1.2 million references, can be found at `http://liinwww.ira.uka.de/bibliography/index.html` and at several mirror sites. The data

[1] The bibliographic databases and support programs for maintaining and manipulating them can be found at `http://www.math.utah.edu:8080/pub/tex/bib/index-table.html`.

included comes from external bibliographical collections like those created by Nelson Beebe. One added-value feature is the search functionality, which allows you to research authors, particular subjects, topics, and other categories. Nearly all of the reference data is available in BIBTEX format.

Another interesting source is CiteSeer, Scientific Literature Digital Library, developed by Steve Lawrence, which can be found at `http://citeseer.nj.nec.com`. Helpful features include extensive search possibilities, context information on publications (e.g., related publications), citations to the document from other publications, statistical information about citations to a citation, and much more.

These examples represent merely a small selection of the vast amount of material found on the Internet. They might prove useful if you are interested in research papers on mathematics, computer science, and similar subjects.

13.4 Bibliography database management tools

As BIBTEX databases are plain text files, they can be generated and manipulated with any editor that is able to write ASCII files. However, with large collections of BIBTEX entries, this method can get quite cumbersome and finding information becomes more and more difficult. For this reason people started to develop tools to help with these tasks. Many of them can be found at `http://www.tug.org/tex-archive/biblio/bibtex/utils/`.

A selection of such tools is described in this section. They range from command-line tools for specific tasks to programs with a graphical user interface for general database maintenance. New products of both types are emerging, so it is probably worthwhile to check out available Internet resources (e.g., `http://bibliographic.openoffice.org/biblio-sw.html`).

13.4.1 biblist—Printing BIBTEX database files

A sorted listing of all entries in a BIBTEX database is often useful for easy reference. Various tools, with more or less the same functionality, are available, and choosing one or the other is mostly a question of taste. In this section we discuss one representative tool, the biblist package written by Joachim Schrod. It can create a typeset listing of (possibly large) BIBTEX databases. Later sections show some more possibilities.

To use biblist you must prepare a LATEX document using the article class. Options and packages like `twoside`, `german`, or geometry can be added. Given that entries are never broken across columns, it may not be advisable to typeset them in several columns using multicol, however.

The argument of the `\bibliography` command must contain the names of all BIBTEX databases you want to print. With a `\bibliographystyle` command you can choose a specific bibliography style. By default, all bibliography entries in the database will be output. However, if you issue explicit `\nocite` commands (as we did in the example), only the selected entries from the databases will be printed. Internal cross-references via the `crossref` field or explicit `\cite` commands are marked using boxes around the *key* instead of resolving the latter.

13-4-1

(June 19, 2004) tex.bib

References

MR-PQ ....
Frank Mittelbach and Chris Rowley.
The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography?
In Vanoirbeek and Coray [EP92], pages 261–273.

EP92 ....
Christine Vanoirbeek and Giovanni Coray, editors.
EP92—Proceedings of Electronic Publishing, '92, Cambridge, 1992. Cambridge University Press.

```
\usepackage{biblist}
\bibliographystyle
           {alpha}
\nocite{MR-PQ}
\footnotesize
\bibliography{tex}
```

You must run LaTeX, BibTeX, and LaTeX. No additional LaTeX run is necessary, since the cross-references are not resolved to conserve space. For this reason you will always see warnings about unresolved citations.

13.4.2 `bibtools`—A collection of command-line tools

Several sets of interesting BibTeX tools are widely available. The first set was written (mostly) by David Kotz. His tools are collectively available for UN*X systems (or `cygwin` under Windows). You may have to adjust the library path names at the top of the scripts to make them work in your environment.

aux2bib Given an `.aux` file, this perl script creates a portable `.bib` file containing only the entries needed for the particular document. This ability is useful when LaTeX files need to be shipped elsewhere. The script works by using a special BibTeX style file (`subset`) to extract the necessary entries, which means that only standard fields are supported.

bibkey This C-shell script uses the sed, egrep, and awk utilities to prepare the list of all entries having a given string as (part of) their citation key.

Usage: `bibkey` *string file*

Characters in the *string* parameter above that have a special meaning in regular expressions used by either sed or egrep must be escaped with a \ (e.g., \\ for the backslash). Case is ignored in the search. Any valid egrep expression is allowed, including, for example, a search for multiple keys:

```
bibkey 'bgb|zpo' jura.bib
```

looktex Entries containing a given string in a BibTeX database are listed when this C-shell script is run. It is a generalization of the bibkey script, and all comments about that script also apply in this case.

Bibliography files
../EX/jura
July 13, 2003

References

[aschur] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht.* München, 29. edition, 2003.

[bgb] Otto Palandt. *Bürgerliches Gesetzbuch.* Beck Juristischer Verlag, München, 62. edition, 2003.

[bschur] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht.* München, 27. edition, 2002.

[zpo] Adolf Baumbach, Wolfgang Lauterbach, Jan Albers, and Peter Hartmann. *Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen.* München, 59. neubearb. edition, 2002.

Figure 13.1: Output of the program printbib

makebib This C-shell script makes an exportable `.bib` file from a given set of `.bib` files and an optional list of citations.

Usage: `makebib` *bibfile(s) [citekey(s)]*

The output is written to `subset.bib`. If *citekey(s)* is not given, then all references in the *bibfile(s)* are included.

printbib This C-shell script makes a `.dvi` file from a `.bib` file for handy reference. It is sorted by cite key and includes `keyword` and `abstract` fields.

Usage: `printbib` *bibfile(s)*

The file `abstract.dvi` is generated and can be run through a `dvi` driver to be printed. Figure 13.1 shows the output when running this shell script on the database `jura.bib` from page 717.

bib2html This perl script produces an HTML version of one or more BibTeX database files.

Usage: `bib2html` *style* `[-o` *outputfile*`]` *bibfile(s)*

There are several *styles* from which to choose; Figure 13.2 on the facing page was produced using the style `alpha` on the `jura.bib` database. If no *outputfile* is given, the file `bib.html` is used as a default. Instead of generating a listing of a complete database you can use the option `-a` and specify an `.aux` file, in which case a bibliography containing only references from this document is created.

Usage: `bib2html` *style* `[-o` *outputfile*`]` `-a` *auxfile*

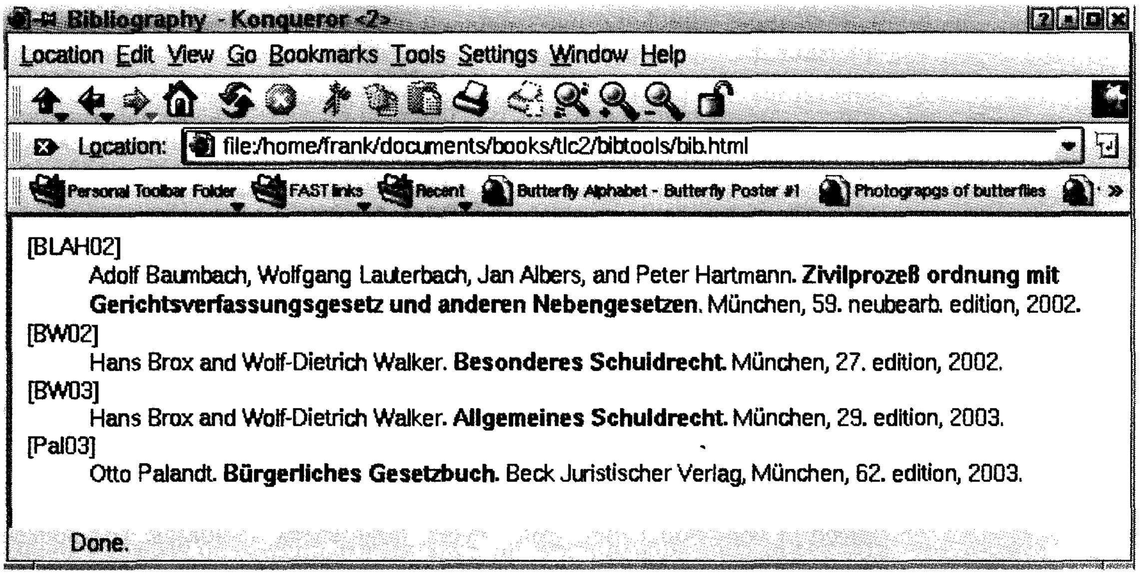


Figure 13.2: Output of the program bib2html

13.4.3 `bibclean`, etc.—A second set of command-line tools

A second set of tools to handle BibTeX databases were developed by Nelson Beebe. We give a brief description of each of them.

bibclean This C program is a pretty-printer, syntax checker, and lexical analyzer for BibTeX bibliography database files [13]. The program, which runs on UN*X, Vax/VMS, and Windows platforms, has many options, but in general you can just type

`bibclean` < *bibfile(s)* > *outfile*

For example, when used on the database file `tex.bib`, the bibclean program reports the following problem:

```
%% "EX/tex.bib", line 92: Unexpected value in ''year = "1980ff"''.
```

bibextract This program extracts from a list of BibTeX files those bibliography entries that match a pair of specified regular expressions, sending them to *stdout*, together with all `@preamble` and `@string` declarations. Two regular expressions must be specified: the first to select `keyword` values (if this string is empty then all fields of an entry are examined), and the second to further select from the value part of the fields which bibliography entries must be output. Regular expressions should contain only lowercase strings.

For example, the following command will extract all entries containing "PostScript" in any of the fields:

`bibextract "" "postscript"` *bibfile(s)* `>` *new-bibfile*

The next command will extract only those entries containing the string `Adobe` in the `author` or `organization` field:

`bibextract "author|organization" "adobe"` *bibfile(s)* `>` *new-bibfile*

Note that one might have to clean the `.bib` files using bibclean before bibextract finds correct entries. For example, the two entries with author "Mittelbach" are found with

```
bibclean tex.bib | bibextract "author" "mittelbach"
```

Using bibextract alone would fail because of the entry containing the line `year=1980ff`.

citefind and citetags Sometimes you have to extract the entries effectively referenced in your publication from several large BibTeX databases. The Bourne shell scripts citefind and citetags use the awk and sed tools to accomplish that task. First, citetags extracts the BibTeX citation tags from the LaTeX source or `.aux` files and sends them to the standard output *stdout*. There, citefind picks them up and tries to find the given keys in the `.bib` files specified. It then writes the resulting new bibliography file to *stdout*. For instance,

`citetags *.aux | citefind -` *bibfile(s)* `>` *outfile*

Nelson Beebe also developed the showtags package, which adds the citation key to a bibliography listing. In other words, it does a similar job to biblist as shown in Example 13-4-1 on page 775 or the program printbib as shown in Figure 13.1 on page 776.

References

MR-PQ

[MR92] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Vanoirbeek and Coray [VC92], pages 261–273.

EP92

[VC92] Christine Vanoirbeek and Giovanni Coray, editors. *EP92—Proceedings of Electronic Publishing, '92*, Cambridge, 1992. Cambridge University Press.

```
\usepackage
   {showtags}
\bibliographystyle
   {is-alpha}
\nocite{MR-PQ}
\footnotesize
\bibliography{tex}
```

13-4-2

13.4.4 bibtool—A multipurpose command-line tool

The program bibtool was developed by Gerd Neugebauer for manipulating BibTeX databases. It combines many of the features from the programs and scripts discussed earlier and adds several new features under the hood of a single program. It is distributed as a C source file, though you may find precompiled binaries—for example, in the Debian distribution. It has been successfully compiled on many architectures, provided a suitable C compiler is available.

In this section we show some of the features provided by the program. Many more are described in the user manual [132] accompanying it.

Pretty-printing, merging, and sorting

In its simplest invocation you can call the program with one or more BibTeX databases as its argument(s), in which case the program acts as a pretty-printer and writes the result to *stdout*.[1] If the option `-o` *file* is used, then the result is written to the specified *file*. For example, to use it on the database shown in Figure 12.2 on page 690, we could write

```
bibtool tex.bib -o new-tex.bib
```

This would produce a pretty-printed version of that database in `new-tex.bib`. All entries will be nicely indented, with every field on a separate line, and all the equals signs will be lined up. For instance, the worst-looking entry in `tex.bib`

```
@manual{GNUMake,     key = {make},
 title = {{GNU Make}, A Program for Directing
 Recompilation},  organization= "Free
 Software Foundation",address = "Boston,
 Massachusetts",ISBN={1-882114-80-9},year = 2000}
```

has now been reformatted as follows:

```
@Manual{          gnumake,
  key           = {make},
  title         = {{GNU Make}, A Program for Directing Recompilation},
  organization  = "Free Software Foundation",
  address       = "Boston, Massachusetts",
  isbn          = {1-882114-80-9},
  year          = 2000
}
```

Merging and sorting

If you specify several database files, then all are merged together in the output. If desired, you can sort them according to the reference keys (using the option `-s` or `-S` for reverse sort). Alternatively, you can specify your own sort key using the resource[2] `sort.format`:

```
bibtool -- 'sort.format="%N(author)"'  tex.bib jura.bib
```

[1] If no input files are specified bibtool reads from *stdin*. Thus, you can also use it as a filter in a UN*X pipe construction, which can be handy sometimes.

[2] Resources are program directives that you assign values. This is often done in external files (explained later); on the command line they can be specified after the -- option.

Be aware that sorting may produce an invalid bibliography file if the file contains internal cross-references, since the entries referenced via a BibTeX `crossref` field have to appear *later* in the database and this may not be the case after sorting. The manual explains how to define sort keys that take this problem into account.

Merging databases together may also result in duplicate entries or, more precisely, in entries that have the same reference keys for use with LaTeX. A database containing such duplicates will produce errors if processed by BibTeX. If you specify the option `-d`, then the duplicates are written out as comments rather than as real entries, which keeps BibTeX happy. However, it might mean that different entries are actually collapsed into a single one (if they happened to have identical keys), so you need to use this option with some care.

Normalization and rewriting of entries

BibTeX supports both double quotes and braces as field delimiters, so the mixture used in the `GNUmake` entry is perfectly legal though perhaps not advisable. A better approach is to stick to one scheme, always using braces or always using double quotes. The rewriting rule

```
bibtool -- 'rewrite.rule {"^\"\([^#]*\)\"$" "{\1}"}'  tex.bib
```

changes the field delimiters to brace groups, except in cases where strings are concatenated. It produces the following result for the sample entry:

```
@Manual{        gnumake,
  key           = {make},
  title         = {{GNU Make}, A Program for Directing Recompilation},
  organization  = {Free Software Foundation},
  address       = {Boston, Massachusetts},
  isbn          = {1-882114-80-9},
  year          = 2000
}
```

Readers who are familiar with regular expressions will probably be able to understand the rather complex field rewriting rule above without further explanation. If not, the manual discusses these features at great length.

Rewriting rules (and, in fact, any other resource definitions) can also be placed in a separate file (default extension `.rsc`) and loaded using the option `-r`. For example, to remove double-quote delimiters you can use

```
bibtool -r braces  tex.bib
```

which loads the distribution file `braces.rsc` containing three rewriting rules similar to the one above covering additional cases.

Rewriting rules can be restricted to work only on certain fields by specifying those fields followed by a # sign before the regular expression pattern. For example, the following rule will rewrite the year field if it contains only two digits potentially surrounded by double quotes or braces and the first digit is not zero (since we do not know if 02 refers to 2002 or 1902):

```
rewrite.rule {year # "^[\"{]?\([1-9][0-9]\)[\"}]?$" "19\1"}
```

Instead of rewriting you can do semantic checks using the `check.rule` resource. For instance, *Semantic checks*

```
check.rule {year # "^[\"{]?\([0-9][0-9]\)[\"}]?$" "\@ \$: year = \1\n"}
```

will generate a warning that a year field with suspicious contents was found if the field contains only two digits (in the message part `\@` is replaced by the entry type and `\$` by the reference key). Applying it to our sample database, we get

```
*** BibTool: Book vleunen:92: year = 92
```

More elaborate semantic checks are discussed in the user manual.

BIBTEX databases may also contain `@string` declarations used as abbreviations in the entries. In certain cases you may want those to be replaced by the strings themselves. This can be done as follows: *Removing `@string` declarations*

```
bibtool -- 'expand.macros=ON' tex.bib
```

This has the result that the series field for the entries `lgc97` and `lwc99` changes from

```
series = ttct
```

to the expanded form

```
series        = {Tools and Techniques for Computer Typesetting}
```

The bibtool program expands strings whose definitions are found in the database files themselves—abbreviations that are part of the BIBTEX style file are left untouched. If they should also get expanded, you have to additionally load a `.bib` file that contains them explicitly as `@string` declarations.

Extracting entries

For selecting a subset of entries from a database a number of possibilities exist. The option `-x` *aux-file* will check in the specified *aux-file* for `\citation` requests and generate from them a new `.bib` file containing only entries required for the particular document. For example:

```
bibtool -x 12-1-1.aux -o 12-1-1.bib
```

There is no need to specify any source database(s), since this information is also picked up from the `.aux` file. Any cross-referenced entries will automatically be included as necessary.

Another possibility is provided with the option `-X` *regexp*, which extracts all entries whose reference key matches the regular expression *regexp*. For example,

```
bibtool -X '^mr-\|^so-' tex.bib
```

will select the two entries with the reference keys `MR-PQ` and `Southall`. Details on regular expressions can be found in the manual. Using regular expressions will select only entries that are explicitly matched. Thus, cross-referenced entries such as `EP92` in this example will not be included automatically, though this outcome can be forced by setting the resource `select.crossrefs` to `ON`.

In addition, several resources can be set to guide selection. For example, to select all entries with Knuth or Lamport as the author or editor, you could say

```
bibtool -- 'select={author editor "Knuth\|Lamport"}' tex.bib
```

To find all entries of type `book` or `article`, you could say

```
bibtool -- 'select={@book @article}' tex.bib
```

To find all entries that do not have a year field, you could say

```
bibtool -- 'select.non={year ".+"}' tex.bib
```

By combining such resource definitions in a resource file and by passing the results of one invocation of bibtool to another, it is possible to provide arbitrarily complicated rewriting and searching methods.

Reference key generation

As we learned in Chapter 12 the reference key, the string used as an argument in the `\cite` command to refer to a bibliography entry, can be freely chosen (with a few restrictions). Nevertheless, it is often a good idea to stick to a certain scheme since that helps you remember the keys and makes duplicate keys less likely. The bibtool program can help here by changing the keys in a database to conform to such a scheme. Of course, that makes sense only for databases not already in use; otherwise, BibTeX would be unable to find the key specified in your documents.

Two predefined schemes are available through the options `-k` and `-K`. They both generate keys consisting of author names and the first relevant word of the title in lowercase (excluding "The" and similar words) and ignoring commands and braces. Thus, when running bibtool on the database from Figure 12.3 on page 717, and then searching for lines containing an @ sign (to limit the listing),

```
bibtool -k jura.bib | grep @
```

we get the following output:

```
@Book{          baumbach.lauterbach.ea:zivilproze,
@Book{          brox.walker:allgemeines,
@Book{          brox.walker:besonderes,
@Book{          palandt:burgerliches,
```

The slightly strange key ending in `:zivilproze` is due to the fact that the entry contains `Zivilproze\ss␣ordnung`, making the program believe the word ends after `\ss`, which itself is discarded because it is a command. Similarly, `\"u` is represented as "u" in the fourth key. You can dramatically improve the situation by additionally loading the resource file `tex_def.rsc`. This file uses the `tex.define` resource to provide translation for common LaTeX commands, so that

```
bibtool -r tex_def -k jura.bib | grep @
```

produces the keys

```
@Book{          baumbach.lauterbach.ea:zivilprozessordnung,
@Book{          brox.walker:allgemeines,
@Book{          brox.walker:besonderes,
@Book{          palandt:buergerliches,
```

Other BIBTEX database-manipulating programs have similar problems in parsing blank-delimited commands, so it is usually better to use `\ss{}` or `{\ss}` in such places. For example, in Figure 13.2 on page 777 you can see that bib2html was also fooled by the notation and added an incorrect extra space in the first entry.

The other key-generating option (`-K`) is similar. It adds the initials of the author(s) after the name:

```
@Book{          baumbach.a.lauterbach.w.ea:zivilproze,
@Book{          brox.h.walker.w:allgemeines,
```

Other schemes can be specified using the powerful configuration options documented in the user manual.

13.4.5 pybliographer—An extensible bibliography manager

The pybliographer scripting environment developed by Frédéric Gobry is a tool for managing bibliographic databases. In the current version it supports the following formats: BibTeX, ISI (web of knowledge), Medline, Ovid, and Refer/EndNote. It can convert from one format to another. It is written in Python, which means that it is readily available on UN*X platforms; usage on Windows systems may prove to be difficult, even though there are Python implementations for this platform as well. The home of pybliographer is `http://pybliographer.org`.

The graphical front end for pybliographer, which builds on the Gnome libraries, is called pybliographic. Upon invocation you can specify a database to work with, usually a local file, though it can be a remote database specified as a URL. For example,

```
pybliographic http://www.math.utah.edu:8080/pub/tex/bib/tugboat.bib
```

will bring up a work space similar to the one shown in Figure 13.3 on the facing page. It will be similar, but not identical, because the graphical user interface is highly customizable. For instance, in the version used by the author an "editor" column was added between "author" and date columns in the main window. If you wish to see other fields use the preference dialog (Settings → Preferences → Gnome). On UN*X systems the preferences are stored in the file `.pybrc.conf`. Although this file is not user editable, you can remove it to restore the default configuration if necessary.

Figure 13.3 shows several other interesting features. On the bottom of the main window you see that the loaded database (`tugboat.bib`) contains 2446 entries, 3 of which are currently displayed. This is due to the fact that we searched it for entries matching the regular expression pattern `Mittelbach` in the `author` field (30 entries found), within the results searched for entries containing `LaTeX3` or `class design` in the `title` field (5 entries found), and within these results restricted the search to publications from the years 1995 to 1999. The search dialog window shows the currently defined hierarchical views available. By clicking on either of them you can jump between the different views; by right-clicking you can delete views no longer of interest. The fields available for searching are customizable. The initial settings offer only a few fields.

To edit an existing entry you can double-click it in the main window. Alternatively, you can use the Edit menu from the toolbar, or you can right-click an entry, which pops up a context menu. The latter two possibilities can also be used to delete entries or add new ones. The edit dialog window shows the entry in a format for manipulation opened at the "Mandatory" tab holding the fields that are mandatory for the current entry type. In addition, there are the optional fields in the "Optional" tab and possibly other fields in the "Extra" tab. This classification is done according to the current settings and can be easily adjusted according to your own preference. While pybliographic is capable of correctly loading databases with arbitrary field names, they will all appear in the Extra tab, which may not be

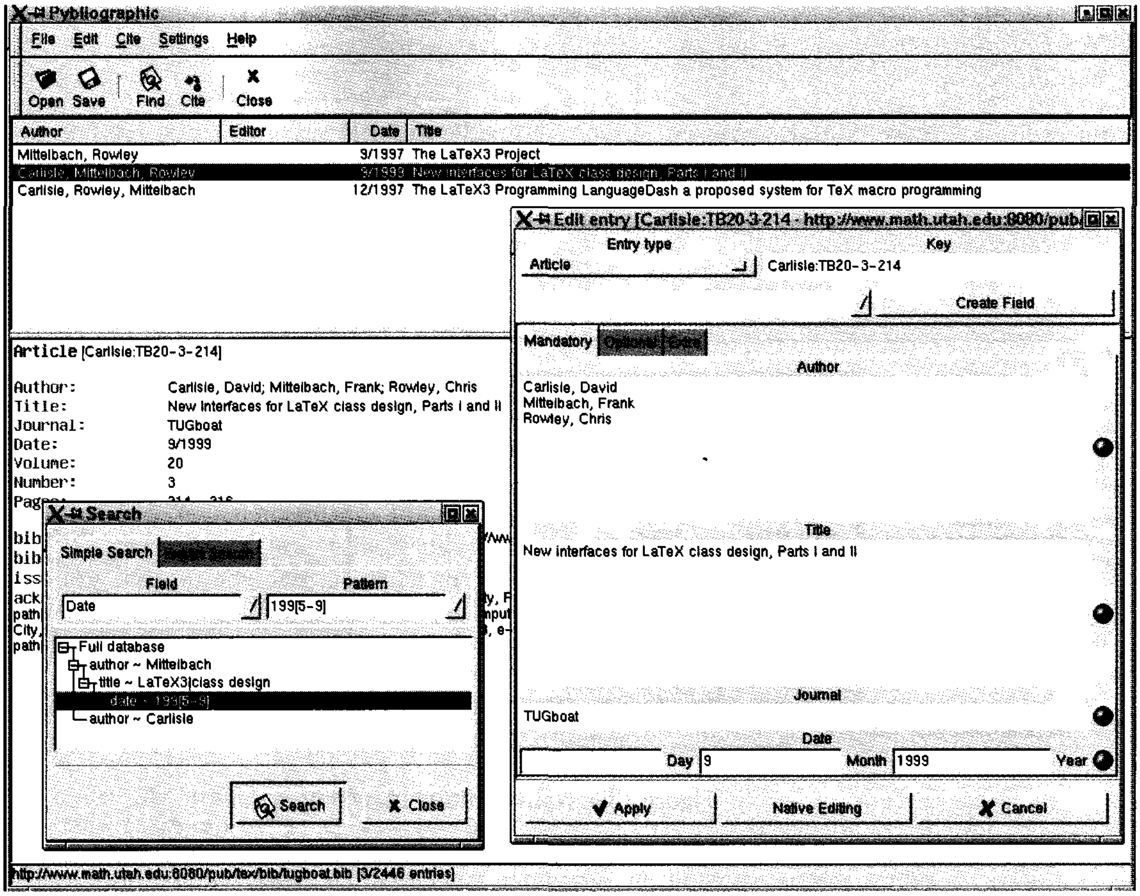


Figure 13.3: The pybliographic work space

convenient if you work with extended BIBTEX styles such as `jurabib` that consider additional fields to be either required or optional. In such cases it pays to adjust the default settings (Settings → Entries, Fields).

Signaling dangerous contents

To the right of the fields you can see round buttons that are either green or red. With the red buttons pybliographic signals that the field content contains some data that the program was unable to parse correctly and that editing the text is likely to result in loss of data. For example, in the `title` field it was unable to interpret the command `\LaTeX{}` correctly and so displayed `LaTeX` instead. The `journal` field is flagged because the database actually contains

```
Journal        = j-tugboat,
```

This reference to an abbreviation would get lost the moment you modify that

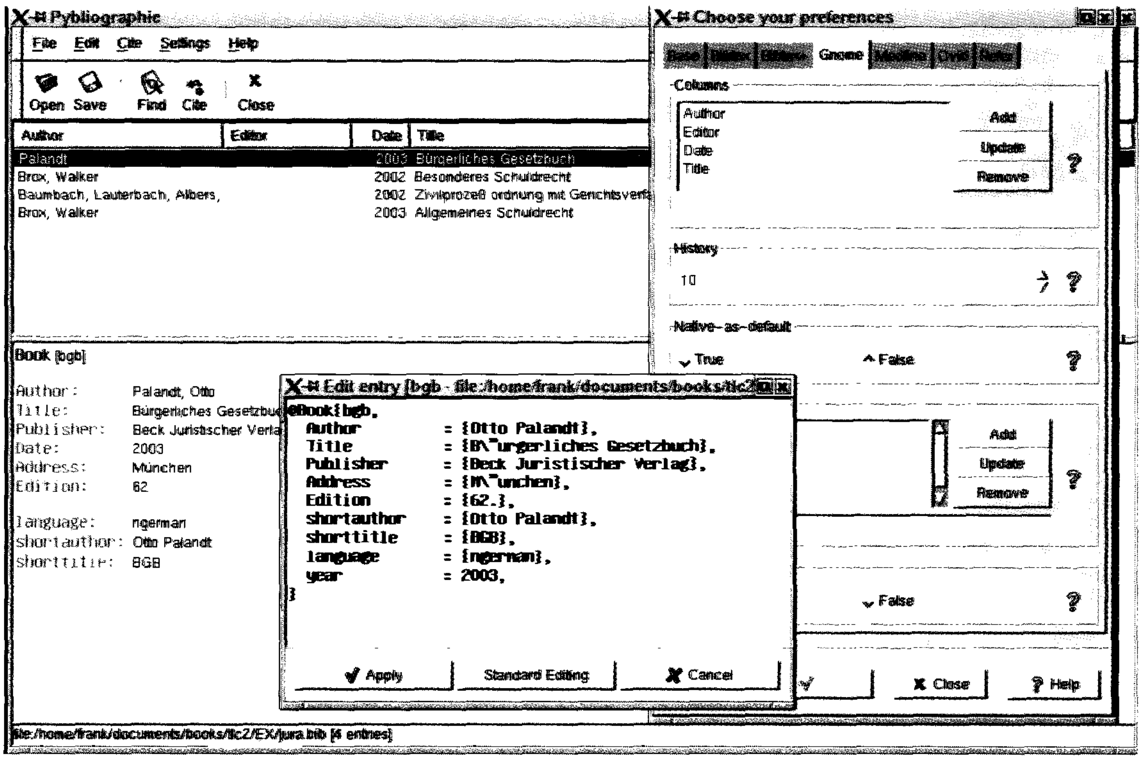


Figure 13.4: Native editing in pybliographic

particular field. To modify such entries you have to change to "Native Editing", as shown in Figure 13.4. This can be done by clicking the "Native Editing" button in the editing dialog window. The window then changes to the format shown in the middle window of Figure 13.4, offering a standard BIBTEX entry format that you can manipulate at will. It is then your responsibility to ensure that the BIBTEX syntax is obeyed. As seen in the right window in that figure, there is the possibility to make the native editing mode serve as the default.

Default capitalization rules

While loading a database pybliographic does some capitalization normalization on a number of fields (e.g., `title`). As this is better done by BIBTEX when formatting for a particular journal you should consider disabling this feature (Settings → Preferences → Bibtex+ → Capitalize). In fact, with languages other than English you have to disable it to avoid proper nouns being incorrectly changed to lowercase.

The distribution also contains a number of command-line scripts. The documentation describes how to provide additional ones. For example, to convert files between different formats you can use pybconvert. The script

```
pybconvert bibtex..refer tex.bib
```

converts the BIBTEX database `tex.bib` to the Refer format, resulting in output such as the following:

```
%T A handbook for scholars
%P xi + 348
%I Oxford University Press
%F vLeunen:92
%D 92
%C Walton Street, Oxford OX2 6DP, UK
%A van Leunen, Mary-Claire
```

Depending on the contents of individual fields you may receive warnings, such as "`warning: unable to convert '\textsl'`", since pybliographer has no idea how to convert such commands to a non-TEX format such as Refer. In that case you should manually correct the results as necessary.

The script pycompact is similar to the aux2bib perl script or the `-x` option of bibtool discussed earlier. However, unlike the latter option, it does not include cross-referenced entries, so it is safer to use bibtool if available.

An interesting script is pybcheck, which expects a list of BIBTEX database files or a directory name as its argument. It then checks all databases for correct syntax, duplicate keys, and other issues. For example, running `pybcheck EX` results in

```
file 'EX/jura.bib' is ok [4 entries]
file 'EX/tex.bib' is ok [12 entries]
```

This script simply verifies the individual databases, so duplicate entries across different files are not detected.

Emacs users can run the command directly from a compile buffer via `M-x compile` followed by `pybcheck` *file(s)*. From the output window you can then jump directly to any error detected using the middle mouse button.

13.4.6 JBibtexManager—A BIBTEX database manager in Java

The JBibtexManager program developped by Nizar Batada is a BIBTEX database manager written in Java; see Figure 13.5 on the following page. Due to the choice of programming language it works on all platforms for which Java 1.4 or higher is available (e.g., Windows, UN*X flavors, Mac).

This program offers searching on the author, editor, title, and keyword values; sorting on the type, reference key, author, year, title, journal, editor, and keywords; and, of course, standard editing functions, including adding, deleting, copying, and pasting between different bibliographies. It automatically detects duplicate reference keys if bibliographies are merged. In addition, it offers the possibility

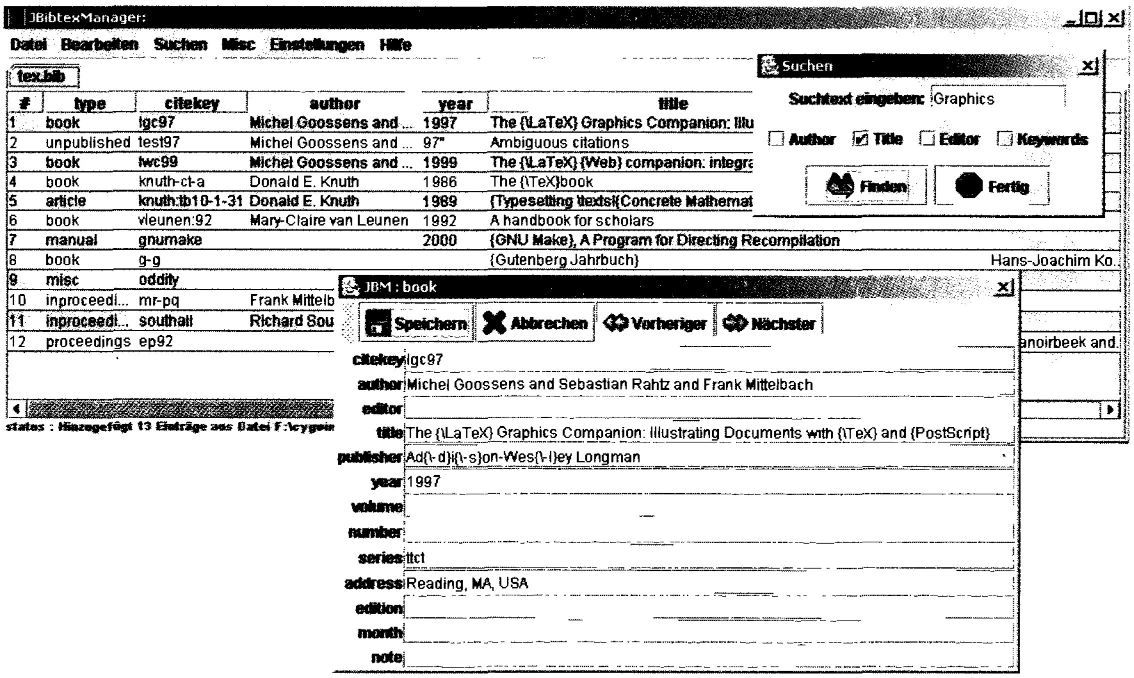


Figure 13.5: The JBibtexManager work space (German locale)

to search a bibliography for duplicate entries (i.e., entries that differ only in their reference keys, if at all).

Like pybliographic, this program can import data in several bibliography formats: BIBTEX, INSPEC, ISI (web of knowledge), Medline (XML), Ovid, and Scifinder. Export formats of HTML and plain text are available. With formats that do not contain any reference key information, the program automatically generates suitable keys provided the author information is structured in a way the program understands.

Although JBibtexManager is intended to work primarily with BIBTEX databases, importing such files for the first time can pose some problems as not all syntax variations of the BIBTEX format are supported. In particular, there should be at most one field per line. Thus, the GNUmake entry in our sample `tex.bib` database would not be parsed correctly. In addition, entries are recognized only if the entry type (starting with the @ sign) starts in the first column. If not, the entry is misinterpreted as a comment and dropped.[1]

Of course, these types of problems happen only the first time an externally generated bibliography is loaded; once the data is accepted by the system, it will

[1]Most of these restrictions have been lifted in the new version of JBibtexManager.

be saved in a way that enables it to be reloaded again. One way to circumvent the problems during the initial loading is to preprocess the external database with a tool like bibtool or bibclean, since after validation and pretty-printing the entries are in an acceptable format.

Unknown fields in a database entry are neither visible nor modifiable except when using the "raw BIBTEX" mode in the newest version of the program. It is, however, possible to customize the recognized fields on a per-type basis so that the program is suitable for use with extended BIBTEX styles such as those used by jurabib or natbib.

The program is available on CTAN. Its current home is `http://csb.stanford.edu/nbatada/JBibtexManager`, but there are plans to merge it with a similar project called BibKeeper under the new name JabRef.

13.4.7 BibTexMng—A BIBTEX database manager for Windows

The BibTexMng program developed by Petr and Nikolay Vabishchevich implements a BIBTEX database manager on Windows; see Figure 13.6 on the next page. It supports all typical management tasks—editing, searching, sorting, moving, or copying entries from one file to another.

In contrast to pybliographic or JBibtexManager, the BibTexMng program deals solely with BIBTEX databases; it has no import or export functions to other bibliographical formats. The only "foreign" export formats supported are `.bbl` files and `.htm` files (i.e., processing a selection of entries with BIBTEX or BIBTEX8 from within the program and producing HTML from a selection of entries).

In the current release the program unfortunately knows about only the standard BIBTEX entry types (see Table 13.1 on page 763), the standard BIBTEX fields (Table 13.2), and the following fields:

`abstract`, `affiliation`, `contents`, `copyright`, `isbn`, `issn`, `keywords`, `language`, `lccn`, `location`, `mrnumber`, `price`, `size`, and `url`

Any other field is silently discarded the first time a BIBTEX database is loaded; the same thing happens to entry types if they do not belong to the standard set. This means that the program is not usable if you intend to work with BIBTEX styles, such as jurabib, that introduce additional fields or types, as neither can be represented by the program. It does, however, work for most styles available, including those intended for natbib (e.g., styles generated with custom-bib).

Not usable with jurabib *et al.*

Another limitation to keep in mind is that the BibTexMng program does not support `@string` declarations. If those are used in an externally generated BIBTEX database, you have to first remove them before using the database with BibTexMng. Otherwise, the entries will be incorrectly parsed. To help with this task the program offers to clean an external database for you (via File → Cleaning of BIBTEX database). This operation replaces all strings by their definitions and removes all unknown fields, if any exist.

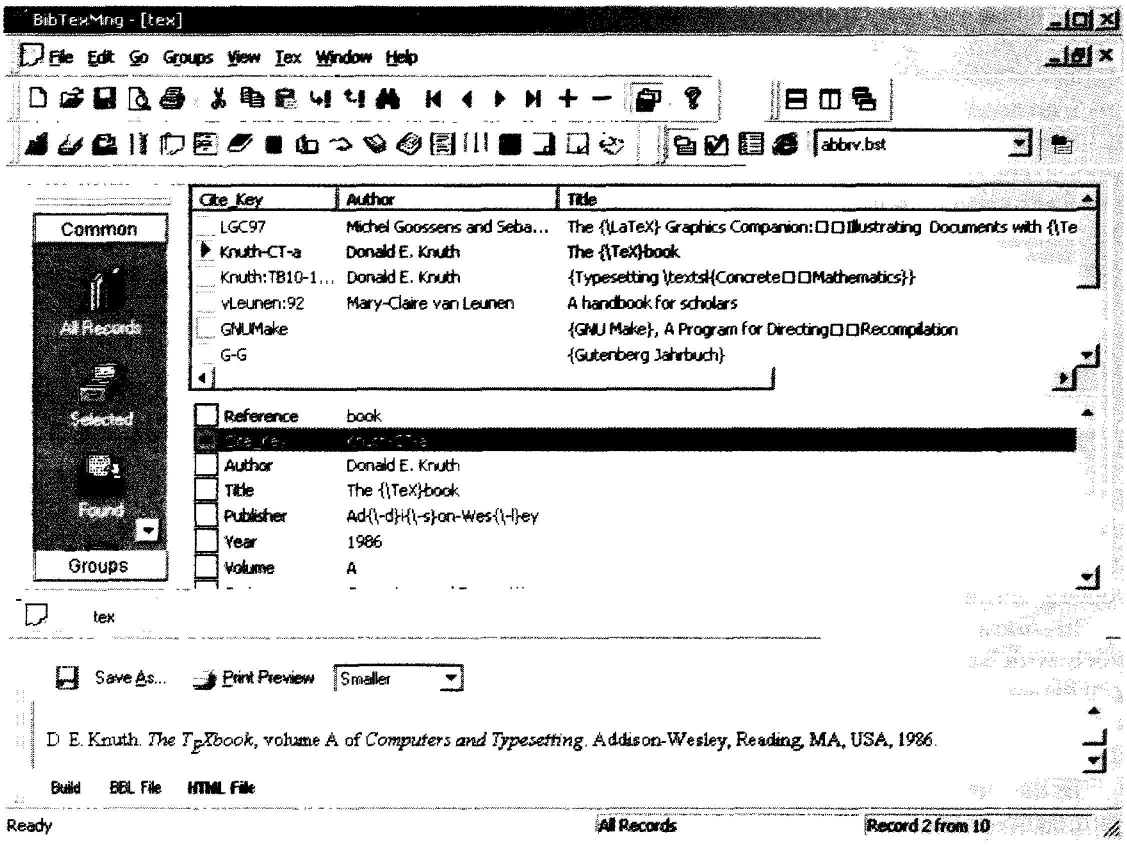


Figure 13.6: The BibTexMng work space

13.5 Formatting the bibliography with BIBTEX styles

Now that we know how to produce BIBTEX database entries and manipulate them using various management tools, it is time to discuss the main purpose of the BIBTEX program. This is to generate a bibliography containing a certain set of entries (determined from the document contents) in a format conforming to a set of conventions.

We first discuss the use of existing styles and present example results produced by a number of standard and nonstandard styles. We then show how the custom-bib package makes it possible to produce customized styles for nearly every requirement with ease.

13.5.1 A collection of BIBTEX style files

Various organizations and individuals have developed style files for BIBTEX that correspond to the house style of particular journals or editing houses. Nelson Beebe has collected a large number of BIBTEX styles. For each style he provides an example file, which allows you to see the effect of using the given style.[1] Some of the BIBTEX styles—for instance, `authordate`⟨*i*⟩, `jmb`, and `named`—must be used in conjunction with their accompanying LATEX packages (as indicated in Table 13.4) to obtain the desired effect.

You can also customize a bibliography style, by making small changes to one of those in the table (see Section 13.6.3 for a description of how this is done). Alternatively, you can generate your own style by using the custom-bib program (as explained in Section 13.5.2 on page 798).

Table 13.4: Selected BIBTEX style files

Style Name	*Description*
`abbrv.bst`	Standard BIBTEX style
`abbrvnat.bst`	natbib variant of `abbrv` style
`abstract.bst`	Modified `alpha` style with `abstract` keyword
`acm.bst`	Association for Computing Machinery BIBTEX style
`agsm.bst`	Australian government publications BIBTEX style
`alpha.bst`	Standard BIBTEX style
`amsalpha.bst`	`alpha`-like BIBTEX style for $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX
`amsplain.bst`	`plain`-like BIBTEX style for $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX (numeric labels)
`annotate.bst`	Modified `alpha` BIBTEX style with `annote` keyword
`annotation.bst`	Modified `plain` BIBTEX style with `annote` keyword
`apa.bst`	American Psychology Association BIBTEX style
`apalike.bst`	Variant of `apa` BIBTEX style
apalike	LATEX package for use with `apalike.bst`
`apalike2.bst`	Variant of `apalike` BIBTEX style
`astron.bst`	*Astronomy* BIBTEX style
`authordate`*i*`.bst`	*i*=[1,4]; series of BIBTEX styles producing author-date reference list
authordate1-4	LATEX package to be used together with `authordate`*i*`.bst`
`bbs.bst`	*Behavioral and Brain Sciences* BIBTEX style
`cbe.bst`	Council of Biology Editors BIBTEX style (includes such journals as *American Naturalist* and *Evolution*)
`cell.bst`	Small modifications to `jmb` BIBTEX style

continued on next page

[1]See Appendix C to find out how you can obtain these files from one of the TEX archives if they are not already on your system.

continued from previous page

Style name	*Description*
harvard	LaTeX package for use with Harvard styles (e.g., `agsm`)
`humanbio.bst`	*Human Biology* BibTeX style
`humannat.bst`	*Human Nature* and *American Anthropologist* journals
`ieeetr.bst`	*Transactions of the Institute of Electrical and Electronic Engineers* BibTeX style
`is-abbrv.bst`	`abbrv` BibTeX style with ISSN and ISBN keyword added
`is-alpha.bst`	`alpha` BibTeX style with ISSN and ISBN keyword added
`is-plain.bst`	`plain` BibTeX style with ISSN and ISBN keyword added
`is-unsrt.bst`	`unsrt` BibTeX style with ISSN and ISBN keyword added
`jmb.bst`	*Journal of Molecular Biology* BibTeX style
jmb	LaTeX package for use with `jmb.bst`
`jox.bst`	Style for use with jurabib (Oxford style)
`jtb.bst`	*Journal of Theoretical Biology* BibTeX style
`jurabib.bst`	Style for use with jurabib
`jureco.bst`	Style for use with jurabib (compact)
`jurunsrt.bst`	Style for use with jurabib (unsorted)
`kluwer.bst`	Kluwer Academic Publishers BibTeX style
`named.bst`	BibTeX style with [author(s), year] type of citation
named	LaTeX package for use with `named.bst`
`namunsrt.bst`	Named variant of `unsrt` BibTeX style
`nar.bst`	*Nucleic Acid Research* BibTeX style
nar	LaTeX package for use with `nar.bst`
`nature.bst`	*Nature* BibTeX style
nature	LaTeX package for use with `nature.bst`
`newapa.bst`	Modification of `apalike.bst`
newapa	LaTeX package for use with `newapa.bst`
`phaip.bst`	*American Institute of Physics* journals BibTeX style
`phapalik.bst`	American Psychology Association BibTeX style
`phcpc.bst`	*Computer Physics Communications* BibTeX style
`phiaea.bst`	Conferences of the International Atomic Energy Agency BibTeX style
`phjcp.bst`	*Journal of Computational Physics* BibTeX style
`phnf.bst`	*Nuclear Fusion* BibTeX style
`phnflet.bst`	*Nuclear Fusion Letters* BibTeX style
`phpf.bst`	*Physics of Fluids* BibTeX style
`phppcf.bst`	Physics version of `apalike` BibTeX style
`phreport.bst`	Internal physics reports BibTeX style
`phrmp.bst`	*Reviews of Modern Physics* BibTeX style

continued on next page

continued from previous page

Style name	*Description*
plain.bst	Standard BIBTEX style
plainnat.bst	natbib variant of plain style
plainyr.bst	plain BIBTEX style with primary sort by year
siam.bst	Society of Industrial and Applied Mathematics BIBTEX style
unsrt.bst	Standard BIBTEX style
unsrtnat.bst	natbib variant of unsrt style

In theory, it is possible to change the appearance of a bibliography by simply using another BIBTEX style. In practice, there are a few restrictions due to the fact that the BIBTEX style interface was augmented by some authors so that their styles need additional support from within LATEX. We saw several such examples in Chapter 12. For instance, all the author-date styles need a special LATEX package such as natbib or harvard to function, and the BIBTEX styles for jurabib will work only if that package is loaded.

On the whole the scheme works quite well, and we prove it in this section by showing the results of applying different BIBTEX styles (plus their support packages if necessary) without otherwise altering the sample document. For this we use the by now familiar database from Figure 12.2 on page 690 and cite five publications from it: an article and a book by Donald Knuth, which will show us, how different publications by the same author are handled; the manual from the Free Software Foundation, which is an entry without an author name; the unpublished entry with many authors and the special BIBTEX string "and others"; and a publication that is part of a proceeding, so that BIBTEX has to include additional data from a different entry.

In our first example we use the standard plain BIBTEX style, which means we use the following input:

```
\bibliographystyle{plain}
\nocite{Knuth:TB10-1-31,GNUMake,MR-PQ,Knuth-CT-a,test97}
\bibliography{tex}
```

To produce the final document, the example LATEX file has to be run through LATEX once to get the citation references written to the .aux file. Next, BIBTEX processes the generated .aux file, reading the relevant entries from the BIBTEX database tex.bib. The actual bibliography style in which the database entries are to be output to the .bbl file for later treatment by LATEX is specified with the command \bibliographystyle in the LATEX source. Finally, LATEX is run twice more—first to load the .bbl file and again to resolve all references.[1] A detailed explanation of this procedure was given in Section 12.1.3 on page 687, where you will also find a graphical representation of the data flow (Figure 12.1).

[1]In fact, for this example only one run is necessary—there are no cross-references to resolve because we used \nocite throughout.

The `plain` style has numeric labels (in brackets) and the entries are alphabetically sorted by author, year, and title. In case of the GNU manual the organization was used for sorting. This will give the following output:

References

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] Michel Goossens, Ben User, Joe Doe, et al. Ambiguous citations. Submitted to the IBM Journal of Research and Development, 1997.

[3] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[4] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

[5] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Christine Vanoirbeek and Giovanni Coray, editors, *EP92—Proceedings of Electronic Publishing, '92*, pages 261–273, Cambridge, 1992. Cambridge University Press. 13-5-1

By replacing `plain` with `abbrv` we get a similar result. Now, however, the entries are more compact, since first names, month, and predefined journal names (Table 13.3 on page 771) are abbreviated. For instance, `ibmjrd` in the second reference now gives "IBM J. Res. Dev." instead of "IBM Journal of Research and Development".

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] M. Goossens, B. User, J. Doe, et al. Ambiguous citations. Submitted to the IBM J. Res. Dev., 1997.

[3] D. E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[4] D. E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, Apr. 1989.

[5] F. Mittelbach and C. Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In C. Vanoirbeek and G. Coray, editors, *EP92—Proceedings of Electronic Publishing, '92*, pages 261–273, Cambridge, 1992. Cambridge University Press. 13-5-2

With the standard BibTeX style `unsrt` we get the same result as with the `plain` style, except that the entries are printed in order of first citation, rather than being sorted. The standard sets of styles do not contain a combination of `unsrt` and `abbrv`, but if necessary it would be easy to integrate the differences between `plain` and `abbrv` into `unsrt` to form a new style.

[1] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

[2] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[3] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Christine Vanoirbeek and Giovanni Coray, editors, *EP92—Proceedings of Electronic Publishing, '92*, pages 261–273, Cambridge, 1992. Cambridge University Press.

[4] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[5] Michel Goossens, Ben User, Joe Doe, et al. Ambiguous citations. Submitted to the IBM
13-5-3 Journal of Research and Development, 1997.

The standard style `alpha` is again similar to `plain`, but the labels of the entries are formed from the author's name and the year of publication. The slightly strange label for the GNU manual is due to the fact that the entry contains a `key` field from which the first three letters are used to form part of the label. Also note the interesting label produced for the reference with more than three authors. The publications are sorted, with the label being used as a sort key, so that now the GNU manual moves to fourth place.

[GUD$^+$97] Michel Goossens, Ben User, Joe Doe, et al. Ambiguous citations. Submitted to the IBM Journal of Research and Development, 1997.

[Knu86] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[Knu89] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

[mak00] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[MR92] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Christine Vanoirbeek and Giovanni Coray, editors, *EP92—Proceedings of Electronic Publishing, '92*, pages
13-5-4 261–273, Cambridge, 1992. Cambridge University Press.

Many BIBTEX styles implement smaller or larger variations of the layouts produced with the standard styles. For example, the `phaip` style for American Institute of Physics journals implements an unsorted layout (i.e., by order of citation), but omits article titles, uses abbreviated author names, and uses a different structure for denoting editors in proceedings. Note that the entry with more than three authors has now been collapsed, showing only the first one.

[1] D. E. Knuth, TUGboat **10**, 31 (1989).

[2] Free Software Foundation, Boston, Massachusetts, *GNU Make, A Program for Directing Recompilation*, 2000.

[3] F. Mittelbach and C. Rowley, The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography?, in *EP92—Proceedings of Electronic Publishing, '92*, edited by C. Vanoirbeek and G. Coray, pages 261–273, Cambridge, 1992, Cambridge University Press.

[4] D. E. Knuth, *The TEXbook*, volume A of *Computers and Typesetting*, Addison-Wesley, Reading, MA, USA, 1986.

[5] M. Goossens et al., Ambiguous citations, Submitted to the IBM J. Res. Dev., 1997. 13-5-5

If we turn to styles implementing an author-date scheme, the layout usually changes more drastically. For instance, labels are normally suppressed (after all, the lookup process is by author). The `chicago` style, for example, displays the author name or names in abbreviated form (first name reversed), followed by the date in parentheses. In addition, we see yet another way to handle the editors in proceedings and instead of the word "pages" we get "pp." For this example we loaded the natbib package to enable author-date support.

Free Software Foundation (2000). *GNU Make, A Program for Directing Recompilation.* Boston, Massachusetts: Free Software Foundation.

Goossens, M., B. User, J. Doe, et al. (1997). Ambiguous citations. Submitted to the IBM Journal of Research and Development.

Knuth, D. E. (1986). *The TEXbook*, Volume A of *Computers and Typesetting*. Reading, MA, USA: Addison-Wesley.

Knuth, D. E. (1989, April). Typesetting Concrete Mathematics. *TUGboat 10*(1), 31–36.

Mittelbach, F. and C. Rowley (1992). The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In C. Vanoirbeek and G. Coray (Eds.), *EP92—Proceedings of Electronic Publishing, '92*, Cambridge, pp. 261–273. Cambridge University Press. 13-5-6

As a final example we present another type of layout that is implemented with the help of the jurabib package. Since more customizing is necessary we show the input used once more. The trick used to suppress the heading is *not* suitable for use in real documents as the space around the heading would be retained!

```
\usepackage[bibformat=ibidem]{jurabib}
\bibliographystyle{jurabib}   \jbuseidemhrule   % use default rule
\renewcommand\refname{}       % suppress heading for the example
\nocite{Knuth:TB10-1-31,GNUMake,MR-PQ,Knuth-CT-a,test97,LGC97}
\bibliography{tex}
```

This will produce a layout in which the author name is replaced by a rule if it has been listed previously. In case of multiple authors the complete list has to be identical (see first two entries). Also, for the first time ISBN and ISSN numbers are shown when present in the entry. If you look closely, you will see many other smaller and larger differences. For example, this is the first style that does not translate titles of articles and proceeding entries to lowercase but rather keeps them as specified in the database (see page 809 for a discussion of how BibTeX styles can be modified to achieve this effect).

As the original application field for jurabib was law citations, it is one of the BibTeX styles that does not provide default strings for the journals listed in Table 13.3 on page 771; as a result, we get an incomplete second entry. BibTeX will warn you about the missing string in this case. You can then provide a definition for it in the database file or, if you prefer, in a separate database file that is loaded only if necessary.

Goossens, Michel/Rahtz, Sebastian/Mittelbach, Frank: The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript. Reading, MA, USA: Addison-Wesley Longman, 1997, Tools and Techniques for Computer Typesetting, xxi + 554, ISBN 0–201–85469–4

Goossens, Michel et al.: Ambiguous citations. 1997, Submitted to the

Knuth, Donald E.: The TeXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

—— Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207

Free Software Foundation: GNU Make, A Program for Directing Recompilation. 2000

Mittelbach, Frank/Rowley, Chris: The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In **Vanoirbeek, Christine/Coray, Giovanni, editors:** EP92—Proceedings of Electronic Publishing, '92. Cambridge: Cambridge University Press, 1992, 261–273

13-5-7

13.5.2 custom-bib—Generate BIBTEX styles with ease

So far, we have discussed how to influence the layout of the bibliography by using different bibliography styles. If a particular BIBTEX style is recommended for the journal or publisher you are writing for, then it is all that it is necessary. However, a more likely scenario is that you have been equipped with a detailed set of instructions that tell you how references should be formatted, but without pointing you to any specific BIBTEX style—a program that may not even be known at the publishing house.

Hunting for an existing style that fits the bill or can be adjusted slightly to do so (see Section 13.6.3) is an option, of course, but given that there are usually several variations in use for each typographical detail, the possibilities are enormous and thus the chances of finding a suitable style are remote. Consider the following nine common requirements for presenting author names:

Requirement	*Example*
Full name surname last	Donald Erwin Knuth/Michael Frederick Plass
Full name surname first	Knuth, Donald Erwin/Plass, Michael Frederick
Initials and surname	D. E. Knuth/M. F. Plass
Surname and initials	Knuth, D. E./Plass, M. F.
Surname and dotless initials	Knuth D E/Plass M F
Surname and concatenated initials	Knuth DE/Plass MF
Surname and spaceless initials	Knuth D.E./Plass M.F.
Only first author reversed with initials	Knuth, D. E./M. F. Plass
Only first author reversed with full names	Knuth, Donald Ervin/Michael Frederick Plass

Table 13.5: Requirements for formatting names

Combining these with a specification for the separation symbol to use (e.g., comma, semicolon, slash), the fonts to use for author names (i.e., Roman, bold, small caps, italic, other), and perhaps a requirement for different fonts for surname and first names, you will get more than 500 different styles for presenting author names in the bibliography. Clearly, this combinatorial explosion cannot be managed by providing predefined styles for every combination.

Faced with this problem, Patrick Daly, the author of natbib, started in 1993 to develop a system that is capable of providing customized BIBTEX styles by collecting answers to questions like the above (more than 70!) and then building a customized `.bst` file corresponding to the answers.

The system works in two phases: (1) a collection phase in which questions are interactively asked and (2) a generation phase in which the answers are used to build the BIBTEX style. Both phases are entirely done by using LATEX and thus can be carried out on any platform without requiring any additional helper program.

The collection is started by running the program `makebst.tex` through LATEX and answering the questions posed to you. Most of the questions are presented in the form of menus that offer several answers. The default answer is marked with a * and can be selected by simply pressing ⟨*return*⟩. Other choices can be selected by typing the letter in parentheses in front of the option. Selecting a letter not present produces the default choice.

Initializing the system

We now walk you through the first questions, which are somewhat special because they are used to initialize the system. Each time we indicate the suggested answer.

```
Do you want a description of the usage? (NO)
```

Replying with y will produce a description of the procedure (as explained above); otherwise, the question has no effect.

```
Enter the name of the MASTER file (default=merlin.mbs)
```

Here the correct answer is ⟨*return*⟩. The default `merlin.mbs` is currently the only production master file available, though this might change one day.

```
Name of the final OUTPUT .bst file? (default extension=bst)
```

Specify the name for your new BibTeX style file, without an extension—for example, `ttct` (Tools and Techniques for Computer Typesetting series). As a result of completing the first phase you will then receive a file called `ttct.dbj` from which the BibTeX style file `ttct.bst` is produced in the second phase.

```
Give a comment line to include in the style file.
Something like for which journals it is applicable.
```

Enter any free-form text you like, but note that a ⟨*return*⟩ ends the comment. It is carried over into the resulting files and can help you at a later stage to identify the purpose of this BibTeX style.

```
Do you want verbose comments? (NO)
```

If you enter y to this question the context of later questions will be shown in the following form:

```
<<STYLE OF CITATIONS:
   ...
>>STYLE OF CITATIONS:
```

Whether this provides any additional help is something you have to decide for yourself. The default is not to provide this extra information.

```
Name of language definition file (default=merlin.mbs)
```

`catalan`	Language support for Catalan	`italian`	Language support for Italian
`dansk`	Language support for Danish	`norsk`	Language support for Norwegian
`dutch`	Language support for Dutch	`polski`	Language support for Polish
`esperant`	Language support for Esperanto	`portuges`	Language support for Portuguese
`finnish`	Language support for Finnish	`slovene`	Language support for Slovene
`french`	Language support for French	`spanish`	Language support for Spanish
`german`	Language support for German		

Table 13.6: Language support in `custom-bib` (summer 2003)

If you are generating a BIBTEX style for a language other than English you can enter the name of the language here. Table 13.6 lists currently supported languages. Otherwise, reply with ⟨*return*⟩.

```
Include file(s) for extra journal names? (NO)
```

By answering y you can load predefined journal names for certain disciplines into the BIBTEX style. You are then asked to specify the files containing these predefined names (with suitable defaults given).

This concludes the first set of questions for initializing the system. What follows are many questions that offer choices concerning layout and functional details. These can be classified into three categories:

Citation scheme The choice made here influences later questions. If you choose author-date support, for example, you will get different questions then if you choose a numerical scheme.

Extensions These questions are related to extending the set of supported BIBTEX fields, such as whether to include a `url` field.

Typographical details You are asked to make choices about how to format specific parts of the bibliographical entries. Several of the choices depend on the citation scheme used.

While it is possible to change your selections in the second phase of the processing (or to start all over again), it is best to have a clear idea about which citation scheme and which extensions are desired before beginning the interactive session. The typographical details can be adjusted far more easily in the second phase if that becomes necessary. We therefore discuss these main choices in some detail.

Selecting the citation scheme

The citation scheme is selected by answering the following question:

```
STYLE OF CITATIONS:
(*) Numerical as in standard LaTeX
```

```
(a) Author-year with some non-standard interface
(b) Alpha style, Jon90 or JWB90 for single or multiple authors
(o) Alpha style, Jon90 even for multiple authors
(f) Alpha style, Jones90 (full name of first author)
(c) Cite key (special for listing contents of bib file)
```

The default choice is "numerical". If you want to produce a style for the author-date scheme, select a (and disregard the mentioning of "nonstandard interface"). For alpha-style citations, use either b, o, or f depending on the label style you prefer. Choice c is of interest only if you want to produce a style for displaying BIBTEX databases, so do not select it for production styles.

If the default (i.e., a numerical citation scheme) was selected, the follow-up question reads:

```
HTML OUTPUT (if non author-year citations)
(*) Normal LaTeX output
(h) Hypertext output, in HTML code, in paragraphs
(n) Hypertext list with sequence numbers
(k) Hypertext with keys for viewing databases
```

Select the default. All other choices generate BIBTEX styles that produce some sort of HTML output (which needs further manipulation before it can be viewed in browsers). This feature is considered experimental.

If you have selected an author-date citation scheme (i.e., a), you will be rewarded with a follow-up question for deciding on the support interface from within LATEX:

```
AUTHOR--YEAR SUPPORT SYSTEM (if author-year citations)
(*) Natbib for use with natbib v5.3 or later
(o) Older Natbib without full authors citations
(l) Apalike for use with apalike.sty
(h) Harvard system with harvard.sty
(a) Astronomy system with astron.sty
(c) Chicago system with chicago.sty
(n) Named system with named.sty
(d) Author-date system with authordate1-4.sty
```

The default choice, natbib, is usually the best, offering all the possibilities described in Sections 12.3.2 and 12.4.1. The option o should *not* be selected. If you have documents using citation commands from, say, the harvard package (see Example 12-3-4 on page 700), the option h would be suitable. For the same reason, the other options might be the right choice in certain circumstances. However, for document portability, natbib should be the preferred choice. Note in particular that some of the other packages mentioned in the options are no longer distributed in the mainstream LATEX installation.

Determining the extensions supported

Besides supporting the standard BIBTEX entry types (Table 13.1 on page 763) and fields (Table 13.2), makebst.tex can be directed to support additional fields as optional fields in the databases, so that they will be used if present. Some of these extensions are turned off by default, even though it makes sense to include them in nearly every BIBTEX style file.

```
LANGUAGE FIELD
(*) No language field
(l) Add language field to switch hyphenation patterns temporarily
```

Replying with l will greatly help in presenting foreign titles properly. Example 12-5-6 on page 719 shows the problems that can arise and explains how they can be resolved when a language field is present (see Example 12-5-36 on page 734). So a deviation from the default is suggested.

```
ANNOTATIONS:
(*) No annotations will be recognized
(a) Annotations in annote field or in .tex file of citekey name
```

Choosing a will integrate support for an annote field in the .bst file as well as support for including annotations stored in files of the form ⟨*citekey*⟩.tex. However, in contrast to jurabib, which also offers this feature, the inclusion cannot be suppressed or activated using a package option. Since you are quite likely to want this feature turned on and off depending on the document, you might be better served by using two separate BIBTEX styles differing only in this respect.

The nonstandard field eid (electronic identifier) is automatically supported by all generated styles. The fields doi, isbn, and issn are included by default but can be deselected. Especially for supporting the REVTEX package from the American Physical Society, a number of other fields can be added.

Finally, support for URLs can be added by answering the following question with something different from the default.

```
URL ADDRESS: (without REVTeX fields)
(*) No URL for electronic (Internet) documents
(u) Include URL as regular item block
(n) URL as note
(l) URL on new line after rest of reference
```

We suggest including support for URLs as references to electronic resources become more and more common. In the bibliography the URL is tagged with \urlprefix\url[*field-value*], with default definitions for both commands. By loading the url package, better line breaking can be achieved.

As one of the last questions you are offered the following choice:

```
COMPATIBILITY WITH PLAIN TEX:
(*) Use LaTeX commands which may not work with Plain TeX
(t) Use only Plain TeX commands for fonts and testing
```

We strongly recommend retaining the default! LaTeX 2_ε is nearly a decade old, and NFSS should have found its way into every living room. Besides, the plain TeX commands (`\rm`, `\bf`, and so on) are no longer officially part of LaTeX. They may be defined by a document class (for compatibility reasons with LaTeX 2.09)—but then they may not. Thus, choosing the obsolete syntax may result in the BibTeX style not functioning properly in all circumstances.[1]

Note that the questions about the extensions are mixed with those about typographical details and do not necessarily appear in the order presented here.

Specifying the typographical details

The remaining questions (of which there are plenty) concern typographical details, such as formatting author names, presenting journal information, and many more topics. As an example we show the question block that deals with the formatting of article titles:

```
TITLE OF ARTICLE:
(*) Title plain with no special font
(i) Title italic (\em)
(q) Title and punctuation in single quotes (`Title,' ..)
(d) Title and punctuation in double quotes (``Title,'' ..)
(g) Title and punctuation in guillemets (<<Title,>> ..)
(x) Title in single quotes (`Title', ..)
(y) Title in double quotes (``Title'', ..)
(z) Title in guillemets (<<Title>>, ..)
```

If you make the wrong choice with any of them, do not despair. You can correct your mistake in the second phase of the processing as explained below.

Generating the BibTeX style from the collected answers

The result of running `makebst.tex` through LaTeX and answering all these questions is a new file with the extension `.dbj`. It contains all your selections in a special form suitable to be processed by DOCSTRIP, which in turn produces the final BibTeX style (see Section 14.2 for a description of the DOCSTRIP program). Technically speaking, a BibTeX bibliographic style file master (`merlin.mbs` by default) contains alternative coding that depends on DOCSTRIP options. By choosing

[1]Warning: in older versions the question was "NEW FONT SELECTION SCHEME" and the default was to use the obsolete commands. So be careful.

entries from the interactive menus discussed above, some of this code is activated, thereby providing the necessary customization.

If you specified `ttct` in response to the question for the new `.bst` file, for example, you would now have a file `ttct.dbj` at your disposal. Hence, all that is necessary to generate the final BIBTEX style `ttct.bst` is to run

```
latex ttct.dbj
```

The content of the `.dbj` files generated from the first phase is well documented and presented in a form that makes further adjustments quite simple. Suppose you have answered `y` in response to the question about the title of articles on the previous page (i.e., use double quotes around the title) but you really should have replied with `d` (use double quotes around title and punctuation). Then all you have to do is open the `.dbj` file with a text editor and search for the block that deals with article titles:

```
%-------------------
%TITLE OF ARTICLE:
%   %: (def) Title plain
% tit-it,%: Title italic
% tit-qq,qt-s,%: Title and punctuation in single quotes
% tit-qq,%: Title and punctuation in double quotes
% tit-qq,qt-g,%: Title and punctuation in guillemets
% tit-qq,qt-s,qx,%: Title in single quotes
  tit-qq,qx,%: Title in double quotes
% tit-qq,qt-g,qx,%: Title in guillemets
%-------------------
```

Changing the behavior then entails nothing more than uncommenting the line you want and commenting out the line currently selected:

```
%-------------------
%TITLE OF ARTICLE:
%   %: (def) Title plain
% tit-it,%: Title italic
% tit-qq,qt-s,%: Title and punctuation in single quotes
  tit-qq,%: Title and punctuation in double quotes
% tit-qq,qt-g,%: Title and punctuation in guillemets
% tit-qq,qt-s,qx,%: Title in single quotes
% tit-qq,qx,%: Title in double quotes
% tit-qq,qt-g,qx,%: Title in guillemets
%-------------------
```

After that, rerun the file through LATEX to obtain an updated BIBTEX style.

13.6 The BibTeX style language

This section presents a condensed introduction to the language used in BibTeX style files. The information should suffice if you want to slightly modify an existing style file. For more details, consult Oren Patashnik's original article, "Designing BibTeX Styles" [136].

BibTeX styles use a postfix stack language (like PostScript) to tell BibTeX how to format the entries in the reference list. The language has 10 commands, described in Table 13.7 on page 807, to manipulate the language's objects: constants, variables, functions, the stack, and the entry list.

BibTeX knows two types of functions: built-in functions, provided by BibTeX itself (see Table 13.8 on page 808), and user functions, which are defined using either the `MACRO` or `FUNCTION` command.

You can use all printing characters inside the pair of double quotes delimiting string constants. Although BibTeX, in general, ignores case differences, it honors the case inside a string. Spaces are significant inside string constants, and a string constant cannot be split across lines.

Variable and function names cannot begin with a numeral and may not contain any of the 10 restricted characters shown on page 769. BibTeX ignores case differences in the names of variables, functions, and macros.

Constants and variables can be of type integer or string (Boolean true and false are represented by the integers 1 and 0, respectively).

There are three kinds of variables:

Global variables These are either integer- or string-valued variables, which are declared using an `INTEGERS` or `STRINGS` command.

Entry variables These are integer- or string-valued variables, which are declared using the `ENTRY` command. Each of these variables will have a value for each entry on the list read in a BibTeX database.

Fields These are string-valued, read-only variables that store the information from the database file. Their values are set by the `READ` command. As with entry variables there is a value for each entry.

13.6.1 The BibTeX style file commands and built-in functions

Table 13.7 on page 807 gives a short description of the 10 BibTeX commands. Although the command names appear in uppercase, BibTeX ignores case differences.

It is recommended (but not required) to leave at least one blank line between commands and to leave no blank lines within a command. This convention helps BibTeX recover from syntax errors.

Table 13.8 on page 808 gives a short overview of BibTeX's 37 built-in functions (for more details, see [136]). Every built-in function with a letter in its name ends with a $ sign.

13.6.2 The documentation style btxbst.doc

Oren Patashnik based the standard BIBTEX style files abbrv, alpha, plain, and unsrt on a generic file, btxbst.doc, which is well documented and should be consulted for gaining a detailed insight into the inner workings of BIBTEX styles.

In the standard styles, labels have two basic formatting modes: *alphabetic*, like [Lam84], and *Numeric*, like [34]. References can be ordered in three ways:

Sorted, alphabetic labels Alphabetically ordered, first by citation label, then by author(s) (or its replacement field), then by year and title.

Sorted, numeric labels Alphabetically ordered, first by author(s) (or its replacement field), then by year and title.

Unsorted Printed in the order in which the references are cited in the text.

The basic flow of a style file is controlled by the following command-lines, which are found at the end of the btxbst.doc file:

```
EXECUTE {begin.bib}          % Preamble and \begin{thebibliography}
EXECUTE {init.state.consts} % Initialize the state constants
ITERATE {call.type$}         % Loop over entries producing output
EXECUTE {end.bib}            % Write \end{thebibliography} command
```

These commands are explained in Tables 13.7 and 13.8.

The code of a style file starts with the declaration of the available fields with the ENTRY declaration and the string variables to be used for the construction of the citation label.

Each entry function starts by calling output.bibitem to write \bibitem and its arguments to the .bbl file. Then the various fields are formatted and printed by the function output or output.check, which handles the writing of separators (commas, periods, \newblock's) as needed. Finally, fin.entry is called to add the final period and finish the entry.

Next come some functions for formatting chunks of an entry. There are functions for each of the basic fields. The format.names function parses names into their "First von Last, Junior" parts, separates them by commas, and puts an "and" before the last name (but ending with "et al." if the last of multiple authors is "others"). The format.authors function applies to authors, and format.editors operates on editors (it appends the appropriate title: ", editor" or ", editors").

The next part of the file contains all the functions defining the different types accepted in a .bib file (i.e., functions like article and book). These functions actually generate the output written to the .bbl file for a given entry. They must precede the READ command. In addition, a style designer should provide a function default.type for unknown types.

`ENTRY` {*field-list*} {*integer-variable-list*} {*string-variable-list*}
Declares the fields and entry variables. BibTeX declares automatically one supplementary field `crossref`, used for cross-referencing, and an additional string entry variable `sort.key$`, used by the `SORT` command. There should be only one `ENTRY` command per style file. For instance, for the styles `alpha` and `plain` you have, respectively,
`ENTRY { address author booktitle ... } {} { label extra.label sort.label }`
`ENTRY { address author booktitle ... } {} { label }`	

`EXECUTE` {*function-name*} Executes a single function. `EXECUTE {begin.bib}`	

`FUNCTION` {*function-name*} {*definition*}
Defines a new function. You cannot change the definition of a `FUNCTION` outside a style file.
`FUNCTION {end.bib}`
`{ newline$ "\end{thebibliography}" write$ newline$ }`	

`MACRO` {*macro-name*} {*definition*} Defines a string macro. You can change the definition of a `MACRO` outside a style file. `MACRO {feb} {"February"}`	
`INTEGERS` {*global-integer-variable-list*} Declares global integer variables. `INTEGERS { longest.label.width last.extra.num }`	
`STRINGS` {*global-string-variable-list*} Declares global string variables. `STRINGS { longest.label last.sort.label next.extra }`	
`ITERATE` {*function-name*} Executes a single function, once for each entry in the list, in the list's current order. `ITERATE {longest.label.pass}`	
`REVERSE` {*function-name*} Executes a single function, once for each entry in the list, in reverse order. `REVERSE {reverse.pass}`	
`READ` Extracts from the database file the field values for each entry in the list. There should be only one `READ` command per style file. The `ENTRY` and `MACRO` commands must precede `READ`.	
`SORT` Sorts the entry list using the values of the string entry variable `sort.key$`.	

Table 13.7: BibTeX style file commands

$\mathcal{I}_1$ $\mathcal{I}_2$	>	($\mathcal{I}$)	1 (if $\mathcal{I}_1$>$\mathcal{I}_2$) or 0 (otherwise)
$\mathcal{I}_1$ $\mathcal{I}_2$	<	($\mathcal{I}$)	1 (if $\mathcal{I}_1$<$\mathcal{I}_2$) or 0 (otherwise)
$\mathcal{I}_1$ $\mathcal{I}_2$	`=`	($\mathcal{I}$)	1 (if $\mathcal{I}_1$=$\mathcal{I}_2$) or 0 (otherwise)
S_1 S_2	`=`	($\mathcal{I}$)	1 (if S_1=S_2) or 0 (otherwise)
$\mathcal{I}_1$ $\mathcal{I}_2$	+	($\mathcal{I}_1$+$\mathcal{I}_2$)	Add two integers
$\mathcal{I}_1$ $\mathcal{I}_2$	−	($\mathcal{I}_1$-$\mathcal{I}_2$)	Subtract two integers
S_1 S_2	*	($S_1 S_2$)	Concatenate two strings
$\mathcal{L}$ $\mathcal{V}$	:=		Assign to $\mathcal{V}$ the value of $\mathcal{L}$
S	`add.period$`	(S.)	Add dot to string unless that string ends with '.', '?', or '!'
	`call.type$`		Execute function whose name is the type of an entry (e.g., `book`)
S `"t"`	`change.case$`	(S)	Convert S to lowercase except at beginning
S `"l"`	`change.case$`	(S)	Convert S completely to lowercase
S `"u"`	`change.case$`	(S)	Convert S completely to uppercase
S	`chr.to.int$`	($\mathcal{I}$)	Translate single string character to ASCII equivalent
	`cite$`	(*cite-string*)	Push `\cite` command argument
$\mathcal{L}$	`duplicate$`	($\mathcal{L}$ $\mathcal{L}$)	Duplicate entry
$\mathcal{L}$	`empty$`	($\mathcal{I}$)	1 (if $\mathcal{L}$ missing field or blank string) or 0 (otherwise)
S_1 $\mathcal{I}$ S_2	`format.name$`	(S)	Format $\mathcal{I}$ names S_1 according to name specifications S_2
$\mathcal{I}$ $\mathcal{F}_1$ $\mathcal{F}_2$	`if$`		Execute $\mathcal{F}_1$ if $\mathcal{I}$>0, else execute $\mathcal{F}_2$
$\mathcal{I}$	`int.to.chr$`	(S)	Translate integer into one ASCII character table
$\mathcal{I}$	`int.to.str$`	(S)	Push string equivalent of integer
$\mathcal{L}$	`missing$`	($\mathcal{I}$)	1 (if $\mathcal{L}$ missing field) or 0 (otherwise)
	`newline$`		Start a new line in the `.bbl` file
S	`num.names$`	($\mathcal{I}$)	Number of names in S
$\mathcal{L}$	`pop$`		Throw away top element on stack
	`preamble$`	(S)	Push concatenation of all `@preamble` strings read in database files
S	`purify$`	(S)	Remove non-alphanumeric characters
	`quote$`	(S)	Push double-quote character string
	`skip$`		Do nothing
	`stack$`		Pop and print whole stack
S $\mathcal{I}_1$ $\mathcal{I}_2$	`substring$`	(S)	Substring of S starting at $\mathcal{I}_1$ and with a length of $\mathcal{I}_2$
$\mathcal{L}_1$ $\mathcal{L}_2$	`swap$`	($\mathcal{L}_2$ $\mathcal{L}_1$)	Swap the literals
S	`text.length$`	($\mathcal{I}$)	Number of "text" characters
S $\mathcal{I}$	`text.prefix$`	(S)	Front $\mathcal{I}$ characters of S
$\mathcal{L}$	`top$`		Pop and print top of stack
	`type$`	(S)	Push current entry's type (e.g., `book` or `""` if unknown)
S	`warning$`		Pop and print top (string) literal and a warning message
$\mathcal{F}_1$ $\mathcal{F}_2$ $\mathcal{I}$	`while$`		Execute $\mathcal{F}_2$ while function value $\mathcal{I}$ of $\mathcal{F}_1$ has $\mathcal{I}$>0
S	`width$`	($\mathcal{I}$)	Push width of S (TeX units)
S	`write$`		Write S to output buffer

Table 13.8: BibTeX style file built-in functions

The built-in functions are preceded by the variable they consume on the stack. If they leave a result on the stack, it is shown in parentheses. A "literal" $\mathcal{L}$ is an element on the stack. It can be an integer $\mathcal{I}$, a string S, a variable $\mathcal{V}$, a function $\mathcal{F}$, or a special value denoting a missing field. If the popped literal has an incorrect type, BibTeX complains and pushes the integer 0 *or the null string, depending on the function's resulting type.*

The next section of the btxbst.doc file contains definitions for the names of the months and for certain common journals. Depending on the style, full or abbreviated names may be used. These definitions are followed by the READ command, which inputs the entries in the .bib file.

Then the labels for the bibliographic entries are constructed. Exactly which fields are used for the primary part of the label depends on the entry type.

The labels are next prepared for sorting. When sorting, the sort key is computed by executing the presort function on each entry. For alphabetic labels you might have to append additional letters (a, b, ...) to create a unique sorting order, which requires two more sorting passes. For numeric labels, either the sorted or the original order can be used. In both cases, you need to keep track of the longest label for use with the thebibliography environment.

Finally, the .bbl file is written by looping over the entries and executing the call.type$ function for each one.

13.6.3 Introducing small changes in a style file

Often it is necessary to make slight changes to an existing style file to suit the particular needs of a publisher.

As a first example, we show you how to eliminate the (sometimes unpleasant) standard BIBTEX style feature that transforms titles to lowercase. In most cases, you will want the titles to remain in the same case as they are typed. A variant of the style unsrt can be created for this purpose. We will call it myunsrt, since it is different from the original style. Similar methods can be used for other styles.

Looking at Table 13.8 on the facing page, you will probably have guessed that function change.case$ is responsible for case changes. With the help of an editor and looking for the above string, you will find that function format.title must be changed. Below we show that function before and after the modification:

```
FUNCTION {format.title}
{
  title empty$
    { "" }
    { title "t" change.case$ }
  if$
}
```

Before Modification

```
FUNCTION {format.title}
{
  title empty$
    { "" }
    { title } % <== modified
  if$
}
```

After Modification

With the help of Table 13.8 on the preceding page, you can follow the logic of the function and the substitution performed.

Another function that must be changed in a similar way is format.edition. Here we can omit the inner if statement since there would be no difference in the branches.

```
FUNCTION {format.edition}
{ edition empty$
   { "" }
   { output.state mid.sentence =
      { edition "l" change.case$
                    " edition" * }
      { edition "t" change.case$
                    " edition" * }
     if$
   }
 if$
}
```

Before Modification

```
FUNCTION {format.edition}
{ edition empty$
   { "" }
   { edition  " edition" * }
  if$
}
```

After Modification

In format.chapter.pages, format.thesis.type, and format.tr.number, similar changes must be made.

Adding a new field

Sometimes you may want to add a new field. As an example, let's add an annote field. Two approaches can be taken: the one adopted in the style annotate or the one used in the style annotation. Let us look at the simpler solution first. The style annotation, based on plain, first adds the field annote to the ENTRY definition list; the fin.entry function is changed then to treat the supplementary field. As seen in the example of the function book, the function fin.entry is called at the end of each function defining an entry type.

```
FUNCTION {fin.entry}
{ add.period$
  write$
  newline$
}
```

Before Modification

```
FUNCTION {fin.entry}
{ add.period$
  write$
  newline$
  "\begin{quotation}\noindent\textsc{Key:\ }" cite$ * write$
    annote missing$
    'skip$
    { "\\\textsc{Annotation:\ }" write$  annote  write$ }
  if$
  "\end{quotation}" write$  newline$
}
```

After Modification

After outputting the citation string inside a quotation environment, the annotation text is written following the text "Annotation", which starts a separate line. If the field is absent, nothing is written (the test, annote missing$, takes the skip$ branch of the if$ command).

The other style, annotate, based on alpha, takes a more complicated approach. After adding the element annotate to the ENTRY definition list, the function format.annotate is created to format that supplementary field. The function has a decision flow similar to the code shown above.

```
FUNCTION {format.annotate}
{ annotate empty$
  { "" }
  { " \begin{quotation}\noindent " annotate * " \end{quotation} " * }
  if$
}
```

The formatting routine for each of the entry types of Table 13.1 on page 763 has a supplementary line `format.annotate write$` just following the call to `fin.entry`.

Foreign language support

If you want to adapt a BIBTEX style to languages other than English, you will, at the very least, have to translate the hard-coded English strings in the BIBTEX style files, like "edition" in the example at the facing page.

First you should edit a style file and introduce the new terms in the necessary places. As you are working with only one language, it is possible to introduce the proper language-specific typographic conventions at the same time. An example of this approach is the `nederlands` style developed by Werenfried Spit. This `harvard`-based style has been adapted to Dutch following the recommendations of Van Dale (1982). We will now look at some examples of functions that were adapted by this style.

In Dutch, one does not distinguish between one or more editors. The generic Dutch word `redactie` replaces the two possibilities.

```
FUNCTION {format.editors}
{ editor empty$
    { "" }
    { editor format.names
      editor num.names$ #1 >
        { " (eds)" * }
        { " (ed.)" * }
      if$
    }
  if$
}
```

Before Modification

```
FUNCTION {format.editors}
{ editor empty$
    { "" }
    {  editor format.names
       ", redactie" *
    }
  if$
}
```

After Modification

The following examples show how, for one particular language, you can go relatively far in the customization (in form and translation) of an entry—in this case, the format of the edition field. In this example, up to the third edition, Dutch-specific strings are used. Starting with the fourth edition, the generic string $\mathtt{i}^{e}$ is used, where `i` is the number of the edition. You can also see the nesting of the `if$` statements and the use of the case-changing command `change.case$`.

```
FUNCTION {format.edition}
{ edition empty$
  { "" }
  { output.state mid.sentence =
    { edition "l" change.case$ " edition" * }
    { edition "t" change.case$ " edition" * }
    if$
  }
  if$
}
```

Before Modification

```
FUNCTION {format.edition}
{ edition empty$
  { "" }
  { edition "1" =
    { "Eerste" }
    { edition "2" =
      { "Tweede" }
      { edition "3" =
        { "Derde" }
        { edition "\textsuperscript{e} " * }
        if$
      }
      if$
    }
    if$
    output.state mid.sentence =
      { "l" change.case$ " druk" * }
      { "t" change.case$ " druk" * }
    if$
  }
  if$
}
```

After Modification

Of course, the strings for the names of the months should be changed and some other language-specific strings can be defined.

```
MACRO {jan} {"januari"}     MACRO {feb} {"februari"}
MACRO {mar} {"maart"}         ...
```

In addition, the sorting routine for the names, `sort.format.names`, must know about the language-dependent rules for showing names in the right order.

Also, most languages have articles or other short words that should be ignored for sorting titles.

```
FUNCTION {sort.format.title}
{ 't :=
  "A " #2
    "An " #3
      "The " #4 t chop.word
    chop.word
  chop.word
  sortify
  #1 global.max$ substring$
}
```

Before Modification

```
FUNCTION {sort.format.title}
{ 't :=
  "De " #3
    "Een " #4 t chop.word
  chop.word
  sortify
  #1 global.max$ substring$
}
```

After Modification

Here the `chop.word` function chops the word specified from the string presented on the stack—in this case, the definite (De) and indefinite (Een) articles.

CHAPTER 14

LaTeX Package Documentation Tools

In this chapter we describe the doc system, a method to document LaTeX macros and environments. A large proportion of the LaTeX code available is documented using its conventions and support tools. The underlying principle is that LaTeX code and comments are mixed in the same file and that the documentation or the stripped package file(s) are obtained from the latter in a standard way. In this chapter we explain the structure that these files should have, and show how, together with the program DOCSTRIP, you can build self-installing procedures for distributing your LaTeX package(s) and generating the associated documentation. This chapter will also help you understand the code written by others, install it with ease, and produce the documentation for it (not necessarily in that order).

We end the chapter with a few words about how version control works and how RCS/CVS information can be extracted with LaTeX. Applying version control methods can be useful for any larger documentation project.

14.1 doc—Documenting LaTeX and other code

The idea of integrated documentation was first employed by Donald Knuth when he developed the TeX program using the WEB system, which combines Pascal-like meta source code and documentation. Thanks to his approach, it was particularly easy to port TeX and its companion programs to practically any computer platform in the world.

Subsequently, authors of LaTeX packages started to realize the importance of documenting their LaTeX code. Many now distribute their LaTeX macros using the framework defined with the doc package (by Frank Mittelbach) and its associated DOCSTRIP utility (originally by Frank Mittelbach with later contributions by Johannes Braams, Denys Duchier, Marcin Woliński, and Mark Wooding). We should mention at this point that there exists an experimental reimplementation with new features and a cleaner and streamlined interface written by Lars Hellström. It is currently distributed as xdoc2, indicating that this is a frozen (and therefore usable) snapshot of work in progress; the final version will be called xdoc.

Both systems allow LaTeX code and documentation to be held in one and the same TeX source file. The obvious advantage is that a sequence of complex TeX instructions becomes easier to understand with the help of comments inside the file. In addition, updates are more straightforward because only a single source file needs to be changed.

The doc package provides a set of commands and establishes some conventions that allow specially prepared sources files to contain both code and its documentation intermixed with each other.

To produce the documentation you need a driver (file) that loads the doc package and then interprets the source file. To produce a ready-to-run version of your code you need to first process the source package with DOCSTRIP (see Section 14.2). This step is usually implicitly done by providing an `.ins` file that is run through LaTeX.

In its simplest form the driver for the documentation is an external file. However, these days the driver is more commonly made part of the source file, so that all you have to do to produce the documentation is to run the source file through LaTeX. The possibilities are discussed in detail in Section 14.1.4.

The most important commands and concepts are discussed in the next sections. Table 14.1 on page 820 gives an overview of all doc user commands. Further details on any of them can be found in the documented source `doc.dtx` of the doc package, which can also serve as a prime (though somewhat aged) example of the doc system. You may additionally want to refer to the tutorial "How to Package Your LaTeX Package" by Scott Pakin, which describes various aspects of the doc package and DOCSTRIP. This tutorial is available on CTAN at `http://www.ctan.org/tex-archive/info/dtxtut`.

14.1.1 General conventions for the source file

A LaTeX file to be used with the doc system consists of *documentation parts* intermixed with *code parts*. Every line of a documentation part starts with a percent sign (%) in the first column. It can contain arbitrary TeX or LaTeX commands, but the % character cannot be used as a comment character. User comments are created by using the `^^A` character instead. Longer text blocks can be turned into comments by surrounding them with `%␣\iffalse` ... `%␣\fi`. All other parts of the file are called code parts. They contain the code described in the documentation parts.

Depending on how the code parts are structured it is possible to use such a file directly with LaTeX, although these days this is seldom done. Instead, DOCSTRIP is typically used to produce the production files. If the former approach is taken LaTeX bypasses the documentation parts at high speed and pastes the macro definitions together, even if they are split into several code parts.

On the other hand, if you want to produce the documentation of the macros, then the code parts should be typeset verbatim. This is achieved by surrounding these parts by the `macrocode` environment.

```
%␣␣␣␣\begin{macrocode}
    ⟨code lines⟩
%␣␣␣␣\end{macrocode}
```

It is mandatory that you put *exactly* four spaces between the `%` character and `\end{macrocode}`. The reason being that when LaTeX is processing the `macrocode` environment, it is actually looking for that particular string and not for the command `\end` with the argument `macrocode`.

Inside a code part all TeX commands are allowed. Even the percent sign can be used to suppress unwanted spaces at the ends of lines.

If you prefer, instead of the `macrocode` environment, you can use the `macrocode*` environment. It produces the same results except that spaces are displayed as ␣ characters when the documentation is printed.

14.1.2 Describing new macros and environments

Most packages contain commands and environments to be employed by users in their documents. To provide a short manual describing their features, a number of constructs are offered by the doc package.

`\DescribeMacro{\`*macro-name*`}`   `\DescribeEnv{`*environment-name*`}`

The `\DescribeMacro` command takes one argument, which will be shown in the margin and produces a special index entry, for example,

```
% \DescribeMacro{\DocInput} \DescribeMacro{\IndexInput}
% Finally the \meta{input commands} part ...
```

A similar macro, `\DescribeEnv`, can be used to indicate that at this point a LaTeX environment is being explained.

`\begin{macro}{\`*macro-name*`}`   `\begin{environment}{`*environment-name*`}`

To describe the definition of a new macro, you use the `macro` environment. It takes one argument: the name of the new macro. This argument is also used to print the name in the margin and to produce an index entry. Actually, the index entries for usage and for definition are different, which allows for easy reference. Here is an example taken from the sources of the doc package itself:

```
% \begin{macro}{\MacroTopsep}
%    Here is the default value for the \verb+\MacroTopsep+
%    parameter used above.
%    \begin{macrocode}
\newlength\MacroTopsep
\setlength\MacroTopsep{7pt plus 2pt minus 2pt}
%    \end{macrocode}
% \end{macro}
```

Another environment, with the unimaginative name `environment`, documents the code of environments. It works like the `macro` environment but expects the name of an environment as its argument.

`\MakeShortVerb{\`*c*`}` `\MakeShortVerb*{\`*c*`}` `\DeleteShortVerb{\`*c*`}`

When you have to quote a lot of material verbatim, such as command names, it is awkward to always have to type `\verb+...+`. Therefore, the doc package provides an abbreviation mechanism that allows you to pick a character *c*, which you plan to use only very rarely inside your document, to delimit your verbatim text (the character " is often chosen, but if that character is already used for another purpose, such as for generating umlauts, then you may prefer "|"). Then, after including the command `\MakeShortVerb{\`*c*`}`, the sequence *c*`text`*c* becomes the equivalent of `\verb`*c*`text`*c*.

The variant form `\MakeShortVerb*` does the same but uses `\verb*`. If you later want to use *c* with its original meaning, just type `\DeleteShortVerb {\`*c*`}`. You can repeat this sequence using *c* as a shorthand for `\verb` and reverting to its original meaning as many times as needed.[1] Note that such short forms for `\verb`, just like `\verb` itself, cannot appear in the argument of another command, but the characters may be used freely inside `verbatim` and `macrocode` environments.

You can divide your documented package file into two parts, the first typically containing a general description and the second giving a detailed description of the implementation of the macros. When generating the document the user will be able to suppress this latter part if you place the command `\StopEventually` at the division point between the two parts.

`\StopEventually{`*final text*`}` `\Finale`

The `\StopEventually` macro takes one argument in which you put all the information that you want to see printed if the user decides to stop typesetting the document at that point (for example, a bibliography, which is usually printed at

[1] This feature has also been made available as a stand-alone package, shortvrb; it was discussed in Section 3.4. See Example 3-4-2 on page 152.

the end of the document). When the driver file contains an `\OnlyDescription` declaration, LaTeX will process the argument of `\StopEventually` and then stop reading the file.[1] Otherwise, the `\StopEventually` macro saves its argument in a macro called `\Finale`, which can later be used to get things back (usually at the very end). This scheme makes changes in two places unnecessary.[2]

To document the change history, the `\changes` command can be placed within the description part of the changed code.

`\changes{`*version*`}{`*date*`}{`*text*`}`

The information in the `\changes` command may be used to produce an auxiliary file (LaTeX's `\glossary` mechanism is used for this purpose), which can be printed after suitable formatting. To cause the change information to be written, include `\RecordChanges` in the driver file. To read and print the sorted change history, put the `\PrintChanges` command at a suitable point, typically after the `\PrintIndex` command in the driver.

To generate the sorted file containing the changes, you should run the raw glossary file through *MakeIndex* using an adequate style (like `gglo.ist`, supplied with the doc distribution; see Section 11.1.6 on page 653 for more information about how *MakeIndex* treats glossaries).

14.1.3 Cross-referencing all macros used

Inside a `macrocode` or `macrocode*` environment, index entries are produced for every command name. In this way you can easily find out where a specific macro is used. Since TeX works considerably more slowly when it has to produce such an array of index entries you can turn off this feature by using `\DisableCrossrefs` in the driver file. To turn it on again, use `\EnableCrossrefs`.

Finer control is provided with the `\DoNotIndex` command, which takes one argument containing a comma-separated list of commands that are *not* to be entered in the index. More than one `\DoNotIndex` command can be present, and their contents will be combined. A frequent use of this macro is to exclude native LaTeX commands from the index.

Production (or not) of index entries is controlled by using or omitting the following declarations in the driver file preamble (if no declaration is provided, no index is produced). Using `\PageIndex` makes all index entries refer to their page number. With `\CodelineIndex`, index entries produced by `\DescribeMacro` and `\DescribeEnv` refer to the relevant page numbers, but those produced by the `macro` and `macrocode` environments refer to the code lines, which are numbered automatically.

[1] The slightly strange command name is due to a misunderstanding by the package author: the German word for "perhaps" is "eventuell" and when he found out it had been in use for years.

[2] The default is to typeset the whole document. This default can also be explicitly set by using the `\AlsoImplementation` macro.

If index entries are produced they have to be sorted by an external program, such as *MakeIndex* (see Chapter 11). The doc package uses special conventions for the index entries, so you need to run *MakeIndex* with the `-s` switch (see Section 11.2.4 on page 659) to specify a suitable style—for example, `gind.ist`, which is distributed with the doc system.

To read and print the sorted index, you must put the `\PrintIndex` command near the end of your driver file, possibly preceded by bibliography commands, as needed for your citations.

14.1.4 The documentation driver

To get the documentation for a set of macros with the doc system, you have to prepare a driver (file) with the following characteristics:

```
\documentclass[⟨options⟩]{⟨document-class⟩}
\usepackage{doc}
⟨preamble⟩
\begin{document}
    ⟨input-commands⟩
\end{document}
```

The ⟨*document-class*⟩ may be any legal class, such as article or ltxdoc (described in Section 14.3); in the latter case the doc package is already loaded by the class. In the ⟨*preamble*⟩, you should place declarations that manipulate the behavior of the doc system, such as `\DisableCrossrefs`, `\OnlyDescription`, and `\CodelineIndex`.

`\DocInput{`*file name*`}` `\IndexInput{`*file name*`}`

Finally, the ⟨*input-commands*⟩ part should contain one or more `\DocInput` and/or `\IndexInput` commands. The `\DocInput` command is used for files prepared for the doc system, whereas `\IndexInput` can be used for macro files that do not obey the conventions of the doc system. The latter command takes a file name as its argument and produces a verbatim listing of the file, indexing every command as it goes along. This functionality can be handy if you want to learn something about macros without enough documentation.

It is also possible to use the `\PrintIndex` and `\PrintChanges` (if the changes are recorded by `\RecordChanges`) commands. Some people put them directly into the source file, but it is better practice to place them into the driver. You can then combine several packages in one document and produce a combined index.

As mentioned in the introduction, most often the driver is included directly in the source file instead of being a separate file of its own. How this works is explained in the next section.

14.1.5 Conditional code in the source

The features discussed so far can be used to produce a LaTeX source in literate programming style that can be directly used by loading it as a package (where TeX bypasses the comments) or printed by processing it with a driver file as explained in the previous section. But this requires the structure of such a file to be linear; in other words, TeX will see *all* code exactly in the order in which it is present in the file.

Experiences with the doc system soon suggested that it would be a valuable extension to be able to conditionally produce the ready-to-run files—by building them from several source files or extracting them from parts of one or more source files, for example. For this reason the doc system was extended in two directions:

- A syntax was developed to label parts of the code so that the components could be referred to separately.
- The DOCSTRIP program (see Section 14.2), which was originally used only to strip the comments from doc files, was extended to offer a scripting language in which it became possible to specify how a ready-to-run file is generated from labeled code parts of one or more source files.

Of course, a source containing such conditional code can usually no longer be used directly and requires the DOCSTRIP program before it can be turned into a ready-to-run file. However, the additional possibilities offered by this approach outweigh the inconvenience of an extra production step during installation so much that these days nearly all usages of doc take advantage of it.

Code fragments for conditional inclusion are marked in the source file with "tags". The simplest format is a <**name*> and </*name*> pair surrounding some part of the code. This enables us to include or exclude that part by referring to its *name* in a DOCSTRIP script. The tags must be placed at the beginning of the line preceded by a %. For example:

```
%<*style>
     some lines of code
%</style>
```

It is possible to attach more than one tag to a part by combining several *names* with the Boolean operators | for logical or, & for logical and, and ! for negation (using lazy evaluation from the left). For example,

```
%<*Aname|Bname&!Cname>
     some lines of code
%</Aname|Bname&!Cname>
```

means that this block should be included when either Aname is asked for, or Bname is requested but Cname is not.

There are two other forms of directives for including or excluding single lines of code. A line starting with `%<+`*name*`>` will be included (without its tag) if *name* is requested. A line starting with `%<-`*name*`>` will be included if *name* is *not* requested in a DOCSTRIP run.

The above directives can be nested in each other. If this is done the inner tags are evaluated only if the outer tags are true (i.e., if the whole block is requested for inclusion).

```
%<*Aname>
        code line 1
%<+Bname> code line 2
%<-Bname> code line 3
        code line 4
%</Aname>
```

Here nothing is included if `Aname` is not requested. If it is requested, we get code lines 1, 2, and 4 if `Bname` is also asked for, and lines 1, 3, and 4 otherwise.

You may have wondered how the conditional coding allows us to include the driver in the main source file. For this you have to place the code for the driver as the first code block and surround it by some tag (e.g., `driver`). If the user now runs the source file through LaTeX, the driver code is the first code that is not behind % signs so it will be executed. Since it ends in `\end{document}`, the LaTeX run will not execute any later code in the file. Thus, the documentation is typeset assuming that the driver loads the whole file using `\DocInput`. To generate the actual package file(s), you use a DOCSTRIP script (see Section 14.2 on page 824) that ignores the driver code by not requesting code from a block tagged `driver`.

Table 14.1: Overview of doc package commands

Preamble and input commands

`\AlsoImplementation`

Typeset complete file marked up according to doc conventions, including code part (default).

`\CharacterTable{`*character table*`}`

User interface to character table checking.

`\CheckModules`

Format module directives of DOCSTRIP specially (default).

`\CheckSum{`*checksum*`}`

User interface to set the checksum of the document (number of backslashes in the code).

`\CodelineIndex`

Index commands using code line numbers.

`\CodelineNumbered`

Number code lines but don't index commands.

continued on next page

continued from previous page

`\DisableCrossrefs`
Don't produce index entries for commands within the code.

`\DocInput{`*file*`}`
Read in *file* assuming `doc` conventions.

`\DontCheckModules`
Don't format module directives of DOCSTRIP specially.

`\EnableCrossrefs`
Produce index entries for commands within the code.

`\IndexInput{`*file*`}`
Read in *file*, print it verbatim, and produce a command cross-reference index.

`\OnlyDescription`
Don't format code; stop at `\StopEventually`.

`\PageIndex`
Index commands using page numbers.

`\PrintChanges`
Print the history listing here.

`\PrintIndex`
Print the index listing here.

`\RecordChanges`
Produce a history listing.

Document structure commands

`\bslash`
Print a backslash (\). Only useful in typewriter fonts!

`\DeleteShortVerb{\`*char*`}`
Undo the previous definition of `\MakeShortVerb` or `\MakeShortVerb*` for *char*.

`\DescribeEnv{`*env*`}`
Flags point in text where environment *env* is described.

`\DescribeMacro{\`*cmd*`}`
Flags point in text where macro *cmd* is described.

`\begin{environment}{`*env*`}`
Environment surrounding description of environment *env*.

`\Finale`
Command executed at very end of document (see also `\StopEventually`).

`\begin{macro}{\`*cmd*`}`
Environment surrounding description of macro *cmd*.

`\begin{macrocode}`
Environment surrounding the TeX code.

`\begin{macrocode*}`
Same as the `macrocode` environment, but spaces are printed as ␣ characters.

`\MakeShortVerb{\`*char*`}`
Define abbreviation character *char* for `\verb`.

continued on next page

continued from previous page

`\MakeShortVerb*{\`*`char`*`}`

Define abbreviation character *char* for `\verb*`.

`\meta{`*`arg`*`}`

Print the argument as a meta sentence (default ⟨*arg*⟩).

`\SpecialEscapechar{\`*`char`*`}`

Specify new single escape character *char* to be used instead of \.

`\StopEventually{`*`cmds`*`}`

In the argument *cmds*, specify which commands should be executed at the end of the document (they are stored in `\Finale`).

`\begin{verbatim}`

Slightly altered version of LaTeX's standard `verbatim` environment to surround verbatim text ignoring percent characters in column 1.

`\begin{verbatim*}`

Same as the `verbatim` environment, but spaces are printed as ␣ characters.

Index commands

`\*`

Symbol used in index entries to refer to a higher-level entry (default ~).

`\actualchar`

Character used to separate "key" and actual index in an index entry (default =).

`\DoNotIndex{`*cmd_1,...,cmd_n*`}`

Names of commands that should not show up in the index.

`\encapchar`

Character used to separate the actual index and the command to format the page number in an index entry (default |).

`\IndexMin`

Length parameter (default 80pt) defining the minimal amount of space that should be left on a page to start an index.

`\IndexParms`

Macro controlling the formatting of the index columns.

`\IndexPrologue{`*`text`*`}`

Overwrite default text to be placed on top of index.

`\levelchar`

Character used to separate different index levels in an index entry (default >).

`\main{`*`number`*`}`

Define the formatting style for page numbers or code line numbers of index entries for major references (default underlined digits).

`\quotechar`

Character used to suppress the special meaning of the following character in an index entry (default !).

`\SortIndex{`*`key`*`}{`*`entry`*`}`

Produce an index entry for *entry*, sorting it by *key*.

continued on next page

continued from previous page

`\SpecialEnvIndex{`*entry*`}`
Produce an index entry for usage of environment *entry*.

`\SpecialIndex{\`*cmd*`}`
Produce a command index (printing the argument verbatim in the index).

`\SpecialMainEnvIndex{`*env*`}`
Produce a main index entry for an environment (`\main` page encapsulator).

`\SpecialMainIndex{\`*cmd*`}`
Produce a main index entry for a macro (`\main` page encapsulator).

`\SpecialUsageIndex{\`*cmd*`}`
Produce an index entry for a macro (`\usage` page encapsulator).

`\usage{`*number*`}`
Define the formatting style for page numbers of index entries for usage descriptions (default italic digits).

`\verbatimchar`
Character used to delimit `\verb` constructs within an index entry (default +).

History information

`\changes{`*version*`}{`*date*`}{`*reason*`}`
Record history information for use in a history listing.

`\docdate`
By convention holds the date of the most recent documentation update.

`\filedate`
By convention holds the date of the most recent code update.

`\filename`
By convention holds the name of the source file.

`\fileversion`
By convention holds the version number of the source file.

`\GlossaryMin`
Length parameter (default 80pt) defining the minimal amount of space that should be left on a page to start the change history.

`\GlossaryParms`
Macro controlling the formatting of the change history columns.

`\GlossaryPrologue{`*text*`}`
Overwrite default text placed on top of history listing.

Layout and typesetting parameters

`\@idxitem`
Macro specifying how index items should be typeset (by default, they are set as a paragraph with a hanging indentation of 30pt for items requiring more than one line).

`\AltMacroFont`
Font used to typeset DOCSTRIP module code (default `\small\ttfamily\slshape`).

`\DocstyleParms`
Macro controlling the formatting of the TeX code.

continued on next page

continued from previous page

`\generalname`
String placed before change entries on the top level.

`\MacrocodeTopsep`
Vertical space above and below each `macrocode` environment.

`\MacroFont`
Font used to typeset the main part of the code (default `\small\ttfamily`).

`\MacroIndent`
Width of the indentation for every code line.

`\MacroTopsep`
Vertical space above and below each `macro` environment.

`\MakePercentComment`
Activate "%" as TeX's comment initiator character.

`\MakePercentIgnore`
Deactivate "%" as TeX's comment initiator character.

`\MakePrivateLetters`
Macro specifying symbols to be considered as letters (default `@`).

`\Module`
Macro with one argument defining the formatting of DOCSTRIP module directives.

`\PrintDescribeEnv`
Macro with one argument defining the formatting of `\DescribeEnv`.

`\PrintDescribeMacro`
Macro with one argument defining the formatting of `\DescribeMacro`.

`\PrintEnvName`
Like `\PrintDescribeEnv` but for the argument of the `environment` environment.

`\PrintMacroName`
Like `\PrintDescribeMacro` but for the argument of the `macro` environment.

`\ps@titlepage`
Macro specifying page style for the title page of articles bundled in a journal (default `\ps@plain`).

`StandardModuleDepth`
Counter holding the highest level of DOCSTRIP directives, which are still formatted using `\MacroFont`. Deeper-nested directives are formatted using `\AltMacroFont`.

`\theCodelineNo`
Control the typesetting of line numbers (default script-size Arabic numerals).

14.2 docstrip.tex—Producing ready-to-run code

When doc was originally written in the late 1980s, the intention was to provide a "literate programming" environment [81] for LaTeX, in which LaTeX code and documentation were intermixed in the same source file. As it soon turned out, making TeX parse (and then ignore) all the documentation when reading a file added a heavy time penalty.[1] To avoid this problem Frank Mittelbach looked for ways to automatically strip all comments from files written for the doc system.

The problem with any external program developed for such a purpose is that it may or may not be available for the user's operating system and even if available may not be installed. But one program is always available on a system that can run LaTeX: the TeX program itself. To achieve widest portability, the DOCSTRIP program was therefore written in low-level TeX language. Since those early days the program has undergone many revisions that changed its purpose from being a simple stripping device to serving as a fully customizable installation tool—one that is even able to distribute files to the right directories on a target machine. Johannes Braams, Denys Duchier, Marcin Woliński, Mark Wooding, David Carlisle, and others contributed to this metamorphosis; details of the program's evolution can be found in the documented source (which uses literate programming, of course). Here are today's main applications of the DOCSTRIP program:

- Strip a literate programming source of most of its documentation (i.e., the lines that start with a single `%` sign in the first column).
- Build ready-to-run code files by using code from one or more source files and including parts of it according to options specified.
- Automatically install the produced files in the right directories on the target machine if desired, thereby enormously easing the installation of updates or additions to a LaTeX installation.

The last possibility in particular is not very widely known but deserves the attention of a wider audience as it can be set up with relatively little effort.

14.2.1 Invocation of the DOCSTRIP utility

From its first days of existence DOCSTRIP could be run interactively by processing `docstrip.tex` with LaTeX:

```
latex docstrip.tex
```

LaTeX then asks a few questions about how to process a given file. When the user has answered these questions, DOCSTRIP does its job and strips the comments from the source.

However, this method of processing was intended to do nothing more than stripping off comments. With today's sources, which contain conditional code and are intended to be combined to form the final "executable", it is usually no longer appropriate. Instead, the developers of packages typically provide an installation file (by convention having the extension `.ins`) that is used to invoke DOCSTRIP behind the scenes. In this case the user simply says

`latex` *name*.ins

[1] In those days producing a single page with TeX could easily take half a minute or longer.

This results in the generation of all "executables" from the source distribution and optionally installs them in the right places. All standard LaTeX distributions (e.g., base, graphics, and tools) are distributed in this form and so are most contributed packages that are described in this book.

In the next section we discuss how to construct your own installation scripts for DOCSTRIP. Section 14.2.3 then shows how to set up DOCSTRIP for automatically installing the generated files in the right places.

14.2.2 DOCSTRIP script commands

A DOCSTRIP installation script has the following general form:

```
\input docstrip
⟨other DOCSTRIP commands⟩
\endbatchfile
```

It starts by loading the DOCSTRIP code using the TeX primitive `\input` (without braces around the file name), which makes it possible to process such a script with TeX formats other than LaTeX. This is followed by the DOCSTRIP commands that actually do the work of building new files, communicating with the user, and carrying out other necessary tasks. At the very end the script contains `\endbatchfile`. Without that statement DOCSTRIP would display a * prompt while waiting for further input from the user.

Generating new files

The main reason for constructing a DOCSTRIP script is to describe which files should be generated, from which sources, and which optional (tagged) pieces of code should be included. This is done by using `\generate` declarations.

```
\generate{\file{result-file1}{\from{source-file1}{tag-list1}
                              \from{source-file2}{tag-list2}
                                ...
                              \from{source-filen}{tag-listn}}
          ...
          \file{result-filen}{...}
         }
```

Within the argument to `\generate` you specify the *result-file* you want to produce by using a `\file` declaration. The second argument to `\file` contains one or more `\from` commands listing the *source-files* that should be used to build the *result-file*. With each `\from` declaration the second argument specifies the *tag-list* to use with the particular *source-file*. Then only the code pieces tagged with the appropriate tags and all the untagged source pieces from that file are included (see Section 14.1.5 on page 819).

The *source-files* are used in the order specified: first the code from *source-file*$_1$ is included (according to the tag specification), then the code from *source-file*$_2$, and so on. The *tag-lists* in each `\from` command are comma-separated lists and indicate that code with these tags should be included.

With the syntax specification for `\generate` as given above, you can produce one *result-file* from one or more *source-files*. By using `\generate` as often as needed, this is general enough to produce any kind of distribution. It is, however, not very efficient. Suppose you have one large source file from which you want to produce many small files—for example, suppose the source for the doc package, `doc.dtx`, is used to generate `doc.sty`, `shortvrb.sty`, `gind.ist`, and `gglo.ist`. The file is nearly 5000 lines long, so by using four `\generate` declarations, DOCSTRIP would have to process 20000 lines. To speed up this process, `\generate` allows you to specify several `\file` commands within its argument. These files are processed in parallel, meaning that the source files are opened only once and distribution of source code to *result-files* is done in parallel.

```
\generate{\file{doc.sty}{\from{doc.dtx}{package}}
          \file{shortvrb.sty}{\from{doc.dtx}{shortvrb}}
          \usepostamble\istpost
          \file{gind.ist}{\from{doc.dtx}{gind}}
          \file{gglo.ist}{\from{doc.dtx}{gglo}}}
```

As you can see, certain other commands (`\usepostamble`, for example) are allowed within the argument of the `\generate` command. In the above example this has the effect of replacing the standard postamble with a different one (since the standard postamble will add an `\endinput` to the end of the generated file, something not desirable in a style file for *MakeIndex*).

There are some restrictions with this approach. For instance, DOCSTRIP will complain if the order of source files in one `\file` command conflicts with the order in a different one; the precise rules are discussed in the DOCSTRIP documentation [125]. If that happens, the simplest solution is to use two separate `\generate` declarations.

Restrictions on parallel extraction

Communicating with the user

The DOCSTRIP scripting language offers some limited possibilities for communication with the user. Keep in mind that interactive questions, though sometimes useful, can make an installation process quite cumbersome, so these tools should be used with care.

`\Msg{`*message*`}` `\Ask{`*cmd*`}{`*question*`}`

The `\Msg` command can be used to present a *message* on the terminal; thus, it offers a similar functionality as LaTeX's `\typeout` command. `\Ask` is similar to LaTeX's `\typein` command, with the difference that no trailing space is generated from pressing return in reply to a *question*. This way simple questions can be asked (using a bit of low-level programming). For example:

```
\Ask\answer{Should we continue? (y/n)}
\ifx\answer\y
  % code for ''y'' as answer
\else
  % otherwise
\fi
```

`\ifToplevel{`*code*`}`

You may want to give certain information, or run certain code, only if a DOCSTRIP script is executed on its own, but not if it is called as part of a larger installation (see below). Such information or code can be placed in the argument of an `\ifToplevel` command. For example, all the individual installation scripts from the LaTeX base distribution say what to do with the generated files. But if you use the master installation script `unpack.ins`, the messages in the sub-scripts are suppressed to avoid repeating the same information over and over again.

`\askforoverwritetrue` `\askforoverwritefalse`

Before DOCSTRIP writes its output to a file, it checks whether that operation will overwrite some existing version of this file. If so, the default behavior is to ask the user if overwriting is acceptable. This check can explicitly be turned off (or on if it was turned off) by using the command `\askforoverwritefalse` or `\askforoverwritetrue`, respectively, in the DOCSTRIP script.

`\askonceonly`

Setting `\askforoverwritefalse` in a distribution script may not be the right thing to do, as it essentially means that it is okay to overwrite other people's files, no matter what. However, for large installations, such as the base LaTeX distribution, being asked individually about hundreds of files is not very helpful either. For this reason DOCSTRIP offers the declaration `\askonceonly`. This means that after the first time the script asks the user a question, the user is given an option to have DOCSTRIP assume that all future questions will get a "yes" as the answer. This applies to *all* future questions (manually produced by `\Ask` or generated through a file overwrite).

`\showprogress` `\keepsilent`

For amusement and because in the original implementation everything was so slow, there was a way to direct DOCSTRIP to show its progress when stripping comments and building new files. These days most scripts run in silent mode.

Master installation scripts

In large distributions, such as the LaTeX base distribution, it is convenient to provide individual DOCSTRIP scripts for processing individual parts. For example, `format.ins` generates the main format file `latex.ltx` and its customization files such as `fonttext.cfg`, and `classes.ins` generates the standard classes, such as the files `article.cls` and `report.cls`.

Nevertheless, you do not want to force the user to process a dozen or more installation scripts (30 in case of the LaTeX base distribution). Therefore, DOCSTRIP offers the command `\batchinput`, which enables you to include installation scripts in some master installation script. Do not use `\input` for this purpose, because this command is exclusively reserved for loading the DOCSTRIP code once, as explained above, and is ignored otherwise. Except for the fact that it contains some special handcrafted code at the beginning so that it can be processed using `initex`, the file `unpack.ins` from the base LaTeX distribution is a good example for such a master installation script.

Setting up preambles and postambles

As mentioned earlier DOCSTRIP not only writes selected lines of code to the output files, but also precedes them with a *preamble* and finishes each file with a *postamble*. There are default texts for both operations, but usually a DOCSTRIP script explicitly defines what should be used in these places, such as a copyright notice or your standard disclaimer (see also [108]).

`\preamble`	`\postamble`
⟨*text lines*⟩	⟨*text lines*⟩
`\endpreamble`	`\endpostamble`

The information you want to add to the start of DOCSTRIP's output file should be listed between the `\preamble` and `\endpreamble` commands. Lines that you want to add at the end should be listed between the `\postamble` and `\endpostamble` commands. Everything that DOCSTRIP finds for both the preamble and postamble is written to the output file, but preceded with two `%` characters (or, more exactly, with the current definition of the command `\MetaPrefix`). In general, only straight text should be used, and literal command names should be of the form `\string\foo`. In addition to the user preamble, DOCSTRIP also includes some information about the current file (i.e., its name and the sources from which it was generated). This information is always added unless you use `\nopreamble` (see below) or you sidestep the standard preamble generation (explained in the DOCSTRIP package documentation [125]).

It is also possible to define a number of "named" preambles or postambles and later refer to them when generating files. In fact, this is the usual way to produce the preambles in larger projects.

`\declarepreamble\`*cmd* ⟨*text*⟩ `\endpreamble` `\usepreamble\`*cmd*
`\declarepostamble\`*cmd* ⟨*text*⟩ `\endpostamble` `\usepostamble\`*cmd*

The `\declarepreamble` declaration works like `\preamble` except that it stores the preamble text for later use in `\`*cmd*. To activate such a preamble, `\usepreamble` is called in a DOCSTRIP script. For postambles, the declarations `\declarepostamble` and `\usepostamble` are provided. Examples of them can be found in all DOCSTRIP installation scripts in the distributions of the standard LaTeX components.

`\nopreamble` `\nopostamble`

To fully suppress the writing of a preamble or a postamble, you can use the declarations `\nopreamble` and `\nopostamble`, respectively.

14.2.3 Installation support and configuration

A number of years ago the TeX users community decided on a standard directory structure for TeX installations (TDS), designed to be usable on all platforms for which TeX and LaTeX are available [164]. Since then this standard has further evolved to the point that it is now in use on most major TeX distributions.

To make it easier to integrate new packages into a TDS-conforming installation or to install package upgrades, the DOCSTRIP program was extended so that under certain circumstances it can be directed to automatically install the generated files in the right places in this structure. For this operation to work, the DOCSTRIP scripts must contain certain directives. In addition, the user has to configure the DOCSTRIP program by providing a `docstrip.cfg` file suitable for the installation on the current machine.

`\usedir{`*relative-directory-path*`}`

For the developer of a DOCSTRIP script there is minimal extra work involved: for each generated file its position in the TDS directory tree needs to be known, but this is usually clear for all such files. This place is then specified with `\usedir` as a directory path relative to the TDS root directory in the DOCSTRIP script just before calling the `\generate` command or within the argument to `\generate` before the next `\file` declaration. For most packages, one such `\usedir` declaration is sufficient. For example, the file `format.ins` in the standard LaTeX distribution states

```
\usedir{tex/latex/base}
\generate{\file{latex.ltx}{\from{ltdirchk.dtx}{initex,2ekernel,dircheck}
                           \from{ltplain.dtx}{2ekernel}
                           ...}
          \file{tracefnt.sty}{\from{ltfsstrc.dtx}{package,trace}}
          \file{flafter.sty}{\from{ltoutput.dtx}{flafter}}
          ...}
```

to place the LaTeX format file (and others) in the correct directory. In more complex bundles, files may need to be distributed to different directories depending on their type. For example, the installation script for the jurabib package states

```
\generate{
   \usedir{tex/latex/jurabib}
     \file{jurabib.sty}{\from{jurabib.dtx}{package}}
     \file{dejbbib.ldf}{\from{jurabib.dtx}{german}}
     ...
   \usedir{bibtex/bst/jurabib}
     \file{jurabib.bst}{\from{jurabib.dtx}{jurabst}}
     ...
   \usedir{doc/latex/jurabib}
     \file{jbtest.tex}{\from{jurabib.dtx}{test}}
     ...
   }
```

to generate the files needed by LaTeX in `tex/latex/jurabib`, the BIBTeX styles in `bibtex/bst/jurabib`, test documents in `doc/latex/jurabib`, and so on. By itself, the `\usedir` declaration has no effect: DOCSTRIP still generates files only in the current directory.

To allow DOCSTRIP to make use of such `\usedir` declarations, you have to provide it with a configuration file (`docstrip.cfg`) that contains a declaration for the root directory of your installation and a set of translations to local directories for the paths used in the argument to `\usedir`.

`\BaseDirectory{`*directory*`}`
`\DeclareDir{`*usedir-path*`}{`*local-translation*`}`

The `\BaseDirectory` declaration specifies the absolute path to the root directory of your TeX installation; other paths are then given relative to this starting directory. In addition, you have to provide for each *relative-directory-path* used in the argument of `\usedir` a translation to a local directory. For example, to teach DOCSTRIP the directory structure used by the emTeX distribution, you might have a set of declarations like this:

```
\BaseDirectory{c:/emtex}
\DeclareDir{tex/latex/base}{texinputs/latex}
\DeclareDir{tex/latex/jurabib}{texinputs/latex}
```

Once DOCSTRIP knows about a `\BaseDirectory`, it will attempt to interpret all `\usedir` declarations in its scripts. If it finds one for which it doesn't know a translation to a local directory (through `\DeclareDir`), it will complain and generate the file in the current directory instead. You should then add an appropriate declaration to the `.cfg` file.

Sometimes it is necessary to put some files outside of the base directory, such as when your BibTeX program is on a different disc. In that case use the starred form of `\DeclareDir`, which expects an absolute path name in the second argument. For example:

```
\DeclareDir*{bibtex/bst/jurabib}{d:/bibtex/bst}
```

Installation directories must exist

Since TeX is unable to create new directories, it is a prerequisite that all local directories specified with `\DeclareDir` actually exist. If one of them is not available when you run a DOCSTRIP script, you will receive a TeX error message stating that it cannot write to some file, and asking you to specify a different one.

On a fully TDS-conforming installation, all translations to local directory names are trivial. For example,

```
\BaseDirectory{/usr/local/lib/texmf-local}
\DeclareDir{tex/latex/base}{text/latex/base}
\DeclareDir{tex/latex/jurabib}{tex/latex/jurabib}
\DeclareDir{bibtex/bst/jurabib}{bibtex/bst/jurabib}
```

directs DOCSTRIP to install into a local TDS tree (i.e., `texmf-local`) and not into the main installation tree. You have then to make sure that your local tree is searched first.

`\UseTDS`

To ease the configuration work necessary to describe a TDS-conforming installation, DOCSTRIP offers the declaration `\UseTDS`. It directs the program to use the `\usedir` specifications literally if no explicit `\DeclareDir` declaration is specified. Thus, on most installations, a `\UseTDS` and a `\BaseDirectory` declaration in the `.cfg` file is all that is needed.

Security considerations

By default, DOCSTRIP will generate files only in the current working directory. Even with a configuration file containing a `\BaseDirectory` declaration, it will always write to directories explicitly specified with `\DeclareDir` or, if you use `\UseTDS`, to the appropriate TDS directories below your base directory. It will not overwrite files in other places, though (in these days of *viruses* and other nasty creatures) you should be aware that TeX, as such, is capable of doing so and therefore might pose some security threat. In fact, some implementations (for example, those on the TeX Live CD) will not let TeX write to files specified with absolute path names or to files starting with a period in their name, unless explicitly authorized. For example, on the author's system one has to specify

```
openout_any=r latex jurabib.ins
```

to take advantage of the automatic installation features of DOCSTRIP.

`\maxfiles{`*number*`}` `\maxoutfiles{`*number*`}`

There are two other declarations that you may wish to add to a DOCSTRIP configuration file. On some operating systems there is a limit on the number of files that can be opened by a program. If that is the case you can limit the total number of open files with a `\maxfiles` declaration and the total number of concurrently opened output files with `\maxoutfiles` (TEX itself has a limit of 16). Use these declarations only when necessary.

14.2.4 Using DOCSTRIP with other languages

With some restrictions it is possible to use the DOCSTRIP mechanism to distribute and generate files not intended for a TEX installation. What you have to bear in mind is that DOCSTRIP operates on a line-by-line basis when reading source files. As a result, doing something like unpacking binary files with it is bound to produce unusable files.

Changing the comment character

Furthermore, the use of preambles and postambles is likely to conflict with the syntax requirements of the language for which the file is intended. For example, generating a shell script with a number of lines starting with `%%` is probably not a good idea. This problem can be circumvented by changing the `\MetaPrefix` (which by default produces `\DoubleperCent`). For a shell script, where you probably want a # sign as the comment character, this modification can be a little tricky as TEX regards the # as special. Try

```
\renewcommand\MetaPrefix{\string##}
```

to produce a single hash sign as a `\MetaPrefix`. To return to the default setting, use the following definition:

```
\renewcommand\MetaPrefix{\DoubleperCent}
```

Verbatim copying

Another potential problem to watch out for is DOCSTRIP's standard behavior of stripping away all lines starting with a single percent sign. If your code contains such lines you may want to retain them. This can be achieved by surrounding that block with two special lines as follows:

`%<<`*tag-name*
  (*code lines to be copied verbatim*)
`%`*tag-name*

You can use any *tag-name*. The important point is that this "verbatim" block ends when DOCSTRIP encounters a single line just containing a percent sign followed by *tag-name*. The other important point to note is that the *tag-name* is not used for conditional exclusion or inclusion but only for specifying the block to be copied verbatim. If such a block should be written only in some circumstances, as controlled through the second argument of `\from`, you have to additionally surround it by a set of conditional tags (see Section 14.1.5).

14.3 ltxdoc—A simple LaTeX documentation class

The ltxdoc class was designed for documenting the core LaTeX source files, which are used to build the LaTeX format and all packages distributed as part of the core distribution. This class is built on the article class, but extends it slightly with a few commands helpful for documenting LaTeX code. It also includes some layout settings specially tailored to accommodate the typical requirements of a source file in doc style (e.g., a line width to hold 72 characters in typewriter fonts and a wider left margin to allow for long macro names to be placed into it).

A special feature is that the class can be used to produce a single document from a larger number of source files in doc style. This has the advantage that one can produce a full index of macro usage across all source files. For example, the driver file `source2e.tex` generates the documented source listing of the 40 files that make up the LaTeX kernel. It generates a document with nearly 600 pages including an index and a change history (reaching back to the early 1990s).

14.3.1 Extensions provided by ltxdoc

As extensions, the class offers a small set of commands to describe LaTeX commands and their arguments. These commands really should have been in the doc package, but due to some historical accident have never been added there.

`\cmd{`*\name*`}` `\cs{`*name*`}`
`\marg{`*arg*`}` `\oarg{`*arg*`}` `\parg{`*arg*`}`

The command `\cmd` prints a command *name* in typewriter font; for example, writing `\cmd{\foo}` typesets `\foo`. In contrast to `\verb+\foo+` (which is otherwise similar), it can be used anywhere—even in the arguments of other commands. The command `\cs` offers the same functionality for those who prefer the syntax without the backslash. In fact, it is slightly more powerful because it can also typeset commands that are made `\outer`—a plain TeX concept normally not used in LaTeX. Furthermore, ltxdoc makes "|" an abbreviation for `\verb` so that you can type `|\foo|` in the documentation. If this is not desired for some reason, you have to cancel it in the source (after `\begin{document}`) via `\DeleteShortVerb{\|}`.

The commands `\marg`, `\oarg`, and `\parg` produce the LaTeX syntax for mandatory, optional, and picture arguments, respectively. Thus, writing

```
\cs{makebox}\parg{x-dimen,y-dimen}\oarg{pos}\marg{text}
```

produces the (probably less-known) syntax diagram for `\makebox` in `picture` environments: `\makebox(`⟨*x-dimen,y-dimen*⟩`)[`⟨*pos*⟩`]{`⟨*text*⟩`}`.

`\DocInclude{`*file*`}`

The `\DocInclude` command is similar to `\include` except that it uses `\DocInput` on *file* (with the implicit extension `.dtx` or `.fdd`) instead of using `\input` on a *file* (with the implicit extension `.tex`). This command is used in `source2e.tex` to "include" all `.dtx` files that form the LaTeX kernel.

14.3.2 Customizing the output of documents that use ltxdoc

To customize documents using the ltxdoc class you can create a configuration file (`ltxdoc.cfg`). This configuration file will be read whenever the ltxdoc class is used, so it can be used to customize the typesetting of all the source files, without having to edit lots of small driver files, which would be the manual alternative.

If `ltxdoc.cfg` is installed in a directory always searched by LaTeX, it is applied to all documentation files using the ltxdoc class. If it is placed in the current directory, it applies only to documents processed in this directory.

The simplest form of customization is to pass one or more options to the article class upon which ltxdoc is based. For instance, if you wish all your documentation to be formatted for A4 paper, add the line

```
\PassOptionsToClass{a4paper}{article}
```

to `ltxdoc.cfg` and install it in a place searched by LaTeX.

As discussed in Section 14.1.2, the `\StopEventually` command separates the source files into a "user" documentation and an "implementation" part. To be able to produce only the user manual, the doc package provides the command `\OnlyDescription`, which suppresses the implementation part. This command may also be used in the configuration file, but as the doc package is loaded after the configuration file is read, you must delay the execution of `\OnlyDescription`. The simplest way is to use `\AtBeginDocument`:

```
\AtBeginDocument{\OnlyDescription}
```

For example, the documented source of the fixltx2e package, the file `fixltx2e.dtx`, generates 30 pages of documented code listings if you run

```
latex fixltx2e.dtx
```

without a configuration file. However, most people are not interested in *how* certain macros from the LaTeX kernel are patched in this package, but rather which problems are solved when loading it. With the above configuration line the output is reduced to a 10-page user manual, listing only the problems that are solved.

When the driver source2e.tex for the kernel documentation is processed, an index and a change history are produced by default; however, indexes are not normally produced for individual files. If you are really interested in the source listings in detail, you will probably want to have an index as well. Again the index commands provided by the doc package may be used, and again their execution must be delayed. Thus, the addition to the configuration file could look as follows:

```
\AtBeginDocument{\AlsoImplementation  % force processing everything
                 \CodelineIndex       % select index per code line
                 \EnableCrossrefs }   % enable it
\AtEndDocument{\PrintIndex}
```

Similar lines would be necessary if you want to produce a change history listing. Recall that the doc package generates .idx and .glo files with a special syntax that require adequate style files for processing with *MakeIndex* (see Section 14.1.3 on page 817).

14.4 Making use of version control tools

When developing a program or writing a large document, such as a user manual or a book (like this one), version control—the task of keeping a software system consisting of many versions and configurations well organized—is an important issue. The *Revision Control System* (RCS) is a software tool that can assist you with that task. RCS manages revisions of text documents—in particular, source programs, documentation, and test data. It automates storage, retrieval, logging, and identification of revisions, and it provides selection mechanisms for composing configurations. In addition, it is able to insert management information in the text document, in so-called *RCS fields*.

The *Concurrent Versions System* (CVS; see http://www.cvshome.org), originally developed as a front end to RCS, extends the notion of revision control from a collection of files in a single directory to a hierarchical collection of directories consisting of revision-controlled files. These directories and files can be combined to form a software release. CVS provides the functions necessary to manage these software releases and to control the concurrent editing of source files among multiple software developers.

RCS and CVS offer a keyword substitution interface in which fields with a certain structure are updated with management information whenever a file is checked into the system. The most important keywords are \$Author\$ (account of the person doing the check-in), \$Date\$ (date and time of check-in in UTC), \$Id\$ (combination field, with file name, revision, date, time, author, state, and optional locked by), \$RCSfile\$ (archive file without path name), \$Revision\$ (revision number assigned to the revision), and \$Source\$ (full path name of archive file). Initially, one simply adds one or more of these keywords (e.g., \$Id\$) to the source.

Upon first check-in, they are replaced by the structure `$`⟨*keyword*⟩`:␣`⟨*value*⟩`␣$`, as can be seen in the next example. Later check-ins then update the ⟨*value*⟩ as appropriate.

If you put LaTeX documents under source control, you will often want to have access to the data of the RCS fields within your document—perhaps to place the date of the last check-in and the revision number into the running heading. Because of the syntax using dollar signs (which indicate formulas in LaTeX), you cannot use the keywords directly in your text, but there exist packages that provide LaTeX tags to give you access to this information in a way suitable for typesetting.

14.4.1 rcs—Accessing individual keywords

The rcs package written by Joachim Schrod lets you extract RCS information from any keyword field and places the data into command names for later use.

`\RCS $`*keyword*`$` `\RCS $`*keyword*`:␣`*value*`␣$`
`\RCSdef $`*keyword*`$` `\RCSdef $`*keyword*`:␣`*value*`␣$`

The `\RCS` command parses a dollar-delimited string for a *keyword* and its corresponding *value*; it is able to recognize the two variants shown above. From the *keyword*, it constructs a command name `\RCS`*keyword* that can be used to later retrieve the *value*. The *keyword* can be any string containing only letters that are usable in a command name; thus, you are not limited to the RCS keyword names mentioned above (though only these keywords are automatically updated by a standard RCS/CVS system). The `\RCSdef` command works like `\RCS` but additionally prints the keyword and value on the terminal.

In the next example we retrieve four typical keys and typeset their values later in the text. As all examples in this book are automatically generated from the book sources (see page 162), the values that you see after the keywords are those corresponding to the file for this chapter.

```
\usepackage{rcs}
\RCS $Date: 2004/08/04 21:57:14 $ \RCS $Author: frank $
\RCS $RCSfile: ch-ldoc.tex,v $   \RCS $Revision: 1.69 $
The file \RCSRCSfile{} has the revision number
\RCSRevision. Last check-in was done by \RCSAuthor{}
on \RCSDate{} at \RCSTime\,\textsc{utc}.
```

14-4-1 The file ch-ldoc.tex,v has the revision number 1.68. Last check-in was done by frank on August 3, 2004 at 20:53:25 UTC.

If you look closely at the previous example, you will notice that `\RCSDate` does not reproduce the value of `$Date$` (which is a numeric date format and the time) but instead produces a date string that looks suspiciously like those being produced by `\today`. This is, in fact, what happens: the value is internally parsed and the check-out date in the format used by `\today` is stored in `\RCSDate`. In this way language-specific packages (e.g., from the babel system) may supply their own methods of presenting a date.

For keywords whose values are further manipulated, the original value is automatically made available in the command \RCSRaw*keyword* (e.g., \RCSRawDate). It is possible to provide your own manipulation routines for other keywords; how this is done is explained in the package documentation (rcs-user.tex).

For convenience, the package defines a couple of additional commands. To parse the $Date$, you can use the command \RCSdate (lowercase "d") instead of the \RCS command used above. This is equivalent to writing

```
\RCS $Date: 2004/08/04 21:57:14 $     \date{\RCSDate}
```

The last check-in date is now automatically used as the date in the document title.[1] Of course, the \RCSDate command is still available for other uses.

Another alternative to \RCS is to use the command \RCSID for parsing a keyword. Besides setting up the corresponding \RCS*keyword* command to hold the value, it typesets the keyword and value literally in the running footer. This command can be used at most once (since each invocation overwrites the footer line) and is best combined with the keyword Id or $Header$. As the rcs package more or less bypasses LaTeX's page style interface, the command does not work if you use \pagestyle commands in your source that update the running footer. In that case use \RCS and manually place the relevant information in the page style using the methods and packages described in Section 4.4.

The package also contains some code to typeset RCS revision history logs that can be produced with the Log keyword. However, this is most likely of no use to the majority of our readers, as it requires a special RCS version and does not work with CVS. If you are interested consult the package documentation.

14.4.2 rcsinfo—Parsing the Id keyword

In contrast to the rcs package, which deals with any string that conforms to the RCS/CVS keyword syntax, the rcsinfo package by Jürgen Vollmer concentrates on a single keyword: Id.

\rcsInfo Id \rcsInfo $Id:␣*value*␣$

If present, the \rcsInfo command parses the *value* and stores all information obtained in a set of commands for later retrieval. Otherwise, it places default values in the retrieval commands—in case of date information, the current date as known to LaTeX and for all other data strings like --owner--.

The following example shows all commands set up by the package and their respective output. As you can see, the \rcsInfoLongDate depends on the current language. Here we get a date in Italian format.

[1] You often see \date{\today} in documents, but this is seldom a good idea because it produces the date of the last formatting run and not the date of the last modification.

ch-ldoc.tex 1.68
2004/08/03 20:53:25
3 agosto 2004
2004, 8, 3 frank
Exp –not-locked–
	`\usepackage[italian]{babel} \usepackage{rcsinfo}`
	`\rcsInfo $Id: ch-ldoc.tex,v 1.69 2004/08/04 21:57:14 frank Exp $`
	`\rcsInfoFile \quad \rcsInfoRevision \par`
	`\rcsInfoDate \quad \rcsInfoTime \par`
	`\rcsInfoLongDate \par`
	`\rcsInfoYear, \rcsInfoMonth, \rcsInfoDay \quad \rcsInfoOwner\par`
	`\rcsInfoStatus \quad \rcsInfoLocker`

14-4-2

To influence its behavior the package offers a few options:

`today/nottoday` By default, `\rcsinfo` changes LaTeX's internal date information to the check-in information obtained. The `\today` command will then generate a date string based on this information. If `nottoday` is used, `\today` will produce a date string showing the date of the LaTeX run.

`fancyhdr/nofancy` When specifying `fancyhdr` the rcsinfo package issues a number of fancyhdr declarations to set up a running footer. You still have to provide your own running header definitions and activate everything with `\pagestyle{fancy}`, so it is probably better to keep full control and do the full set-up yourself.

`long/short` This option works only if the `fancyhdr` option is used. It then decides whether a long (default) or a short date string is used in the footer line.

For those who want to convert their LaTeX documents to HTML using the latex2html program [56, Chapter 3], rcsinfo offers direct support in the form of a perl file, `rcsinfo.perl`; this file must be placed in the appropriate directory in the latex2html installation. Refer to the rcsinfo manual for more information.

Appendix A

A LaTeX Overview for Preamble, Package, and Class Writers

This appendix gives an overview of the basic programming concepts underlying the LaTeX formatter. We explain how to define new commands and environments, including those with an optional argument. We discuss how LaTeX handles counters and their representation; we also introduce horizontal and vertical space parameters and explain how they are handled. The second section reviews the important subject of (LA)TeX boxes and their use. A good understanding of this topic is very important to fully appreciate and exploit the information presented in this book. The third section is devoted to two package files, calc and ifthen, that make calculations and building control structures with LaTeX easier. They have been used in many examples of LaTeX code throughout this book. Finally, we describe in detail the LaTeX 2_ε interface that allows you to define your own options for packages and class files.

A.1 Linking markup and formatting

This section reviews the syntax for defining commands and environments with LaTeX. It is important that you exclusively use the LaTeX constructs described below, rather than the lower-level TeX commands. Then, not only will you be able to take advantage of LaTeX's consistency checking, but your commands will also be portable, (probably) without modification, to future versions of LaTeX.

A.1.1 Command and environment names

Commands

In the current LaTeX incarnation, it is possible to enter accented characters and other non-ASCII symbols directly into the source, so it would seem reasonable to expect that such characters could also be used in command and environment names (e.g., `\größer`). However, this is not the case—LaTeX multi-character command names must be built from basic ASCII letters (i.e., `a...z` and `A...Z`).[1] This means that `\vspace*` is actually not a command by itself; rather, it is the command `\vspace` followed by the modifier `*`. Technically, you could write `\vspace␣*` (as the space is ignored) or even put the `*` on the next line of your document.[2]

Environments

On the other hand, names of environments are different. In this case the `*` is part of the name and spaces preceding it are not ignored. Thus, when writing `\begin{figure␣*}`, the space would become part of the name and is not recognized as the start of a `figure*` environment. This is due to implementation details and seems to indicate that with environment names some additional ASCII characters work. For example:

```
\newenvironment{foo.bar:baz␣with␣space}{}{}
```

However, this is not true in general because, depending on additional packages being loaded, such environment names may no longer be recognized or may produce strange errors. Thus, it is best not to explore that implementation (mis)feature and instead to rely on officially supported names—those containing only lowercase and uppercase letters and the star character.

Citation and label keys

Strictly speaking, `\cite` and `\label` keys have the same kind of restriction. Nevertheless, it has become common practice to use keys containing colons (e.g., `sec:cmds`), so that most packages provide extra support to allow for at least the colon character in such keys. Characters outside the ASCII range and characters used in LaTeX's syntax (e.g., `_` or `#`) can never be used in names, whether they are keys, counters, environments, or multi-character command names.

With single-character command names, the situation is different again: any (single) character can be used. For example, `\$` is a perfectly valid LaTeX command, but `\foo$bar` would be interpreted as the command `\foo` followed by the start of a math formula (signaled by `$`) followed by the (math) characters `b`, `a`, and `r`. Any following text will also be typeset in math mode.

LaTeX commands (i.e., those constructs starting with a backslash) are classified into three basic categories: document-level commands, package and class writer commands, and internal "kernel" commands.

[1] Strictly speaking this is not true, as TeX can be configured to support other configurations. There are, however, valid reasons why this is not being done for standard LaTeX. Some of these reasons are discussed in Section 7.11 describing LaTeX's encoding model.

[2] It is bad style to use this in your documents but there is unfortunately no way to prevent it.

Document-level commands

Document-level commands, such as `\section`, `\emph`, and `\sum`, usually have (reasonably) short names, all in lowercase.

Class and package writer commands

Class and package writer commands, by convention, have longer mixed-case names, such as `\InputIfFileExists` and `\RequirePackage`. Some of them can be usefully applied in the document source, but many will stop working after `\begin{document}` has been processed.

Internal LaTeX commands

Most of the internal commands used in the LaTeX implementation, such as `\@tempcnta`, `\@ifnextchar`, and `\z@` contain `@` in their name. This effectively prevents these names from being used in documents for user-defined commands. However, it also means that they cannot appear in a document, even in the preamble, without taking special precautions.

Careful with internal commands!

As a few of the examples in this book demonstrate, it is sometimes necessary to have such bits of "internal code" in the preamble. The commands `\makeatletter` and `\makeatother` make this easy to do; the difficult bit is to remember to add them, failure to do so can result in some strange errors. For an example of their use, see page 852. Note that package and class files should never contain these commands: `\makeatletter` is not needed as this is always set up when reading such files; and the use of `\makeatother` would prematurely stop this behavior, causing all kinds of havoc.

Unfortunately, for historical reasons the distinction between these categories is often blurred. For example, `\hbox` is an internal command that should preferably be used only in the LaTeX kernel, whereas `\m@ne` is the constant `-1` and could have been `\MinusOne`.

Nevertheless, this rule of thumb is still useful: if a command has `@` in its name, then it is not part of the supported LaTeX language—and its behavior may change in future releases! Any such command should be used with great care. On the other hand, mixed-case commands or those described in the *LaTeX Manual* [104] are guaranteed to be supported in future releases of LaTeX 2_ε.

A.1.2 Defining new commands

It is often advantageous to define new commands (e.g., for representing repetitive input strings or recurring combinations of commands). A new command is defined using the `\newcommand` command sequence, which can have one optional argument, defining the number of arguments taken by the new command.

`\newcommand{`*cmd*`}[`*narg*`]{`*command definition*`}`

The number of arguments is in the range $0 \leq narg \leq 9$. If your new command has no arguments, then the `[0]` can be omitted. Inside the *command definition* part, the arguments are referenced as `#1` to `#`*narg*.

A-1-1

PostScript and its variant Encapsulated PostScript are often used for including graphics in LaTeX documents ...

```
\newcommand{\PS}{Post\-Script}
\newcommand{\EPS}{Encapsulated \PS}
\PS{} and its variant \EPS{} are often used for
including graphics in \LaTeX{} documents \ldots
```

The *cmd* argument always has to contain a single "token" (the name of the command to be defined), so one can omit the braces around this argument. While we do not recommend the use of this TeX syntax feature in other places, it is commonly used with `\newcommand` and similar declarations. In fact, we have often used this more concise syntax in this book:

Omitting argument braces

```
\newcommand\PS {Post\-Script}
\newcommand\EPS{Encapsulated \PS}
```

Note, however, that this is only possible with arguments that are single tokens to TeX (i.e., names starting with a backslash). Trying to do the same with, for instance, environment or counter names will fail. For example,

```
\setcounter mycount {5}
\newenvironment myenv{...}{...}
```

is invalid LaTeX syntax.

If a command should work both in math mode and in text mode, special care should be taken in its definition. One could, for example, use `\mbox` but this has a number of drawbacks.

The series of $x_1,\ldots,x_n$ or $x_1,\ldots,x_n + Gx_1,\ldots,x_n$

```
\newcommand\xvec{\mbox{$x_1,\ldots,x_n$}}
The series of \xvec\ or $\xvec+G_{\xvec}$
```

A-1-2

A better solution is offered by the LaTeX 2_ε command `\ensuremath`. As the name implies, `\ensuremath` ensures that its argument is always typeset in math mode by surrounding it, if necessary, with $ signs. Thus, the definition in the above example should be replaced as follows:

The series of $x_1,\ldots,x_n$ or $x_1,\ldots,x_n + G_{x_1,\ldots,x_n}$

```
\newcommand\xvec{\ensuremath{x_1,\ldots,x_n}}
The series of \xvec\ or $\xvec+G_{\xvec}$
```

A-1-3

This has the additional advantage of producing correctly sized symbols in subscripts or superscripts, which is not the case if an `\mbox` is used in the definition.

Existing commands must be *redefined* with the command `\renewcommand`, which otherwise has the same syntax as `\newcommand`. Note that you can redefine a command with a different number of arguments than the original one has. Therefore, you could redefine the `\xvec` command of the above example, so that it now takes one argument:

The series of $x_1,\ldots,x_n$ or $x_1,\ldots,x_n + G_{x_1,\ldots,x_n}$
The series of $x_1,\ldots,x_n$ or $x_1,\ldots,x_k + G_{x_1,\ldots,x_k}$

```
\newcommand\xvec{\ensuremath{x_1,\ldots,x_n}}
The series of \xvec\ or $\xvec+G_{\xvec}$ \par
\renewcommand\xvec[1]{\ensuremath{x_1,\ldots,x_{#1}}}
The series of \xvec{n} or $\xvec{k}+G_{\xvec{k}}$
```

A-1-4

When redefining a command (or an environment—see below), you must, of course, be cautious. Commands that you are planning to redefine might be used in the class or packages you have loaded (try redefining `\uppercase` in a document that is formatted with the class **book**).

Commands with one optional argument

In LaTeX, you can also define commands so that their first argument is optional. The syntax is

`\newcommand{`*cmd*`}[`*narg*`][`*default*`]{`*command definition*`}`

An example of such a command definition is shown below:

```
\newcommand\LB[1][3]{\linebreak[#1]}
```

The default for the optional argument is given between the second pair of square brackets—the string "3" in this case. Inside the command definition, the optional argument has the number `#1`, while the mandatory arguments (when present) are addressed `#2` to `#`*narg*. Thus, typing `\LB` is a short way of saying `\linebreak[3]`, while `\LB[2]` uses the actual specified value. That is, you will obtain the same effect as when typing `\linebreak[2]`.

In the next example we define the command `\lvec`, which can be used inside or outside of formulas (due to `\ensuremath`). Under the assumption that the upper subscript is usually n we made it optional, while the vector variable has to be given explicitly.

A-1-5

For the series $x_1 + \cdots + x_n$ we have

$$x_1 + \cdots + x_n = \sum_{k=1}^{n} G_{y_1+\cdots+y_k}$$

```
\newcommand\lvec[2][n]
 {\ensuremath{#2_1+\cdots + #2_{#1}}}
For the series \lvec{x}  we have
\[ \lvec{x} = \sum_{k=1}^{n} G_{\lvec[k]{y}} \]
```

In general, it is most practical to associate the case that occurs most often with the form of the command without parameters and to represent the cases that are used less often with longer command strings with an optional argument.

Argument restrictions

As explained above, user-defined commands can have one optional argument and up to nine arguments in total. If defined with `\newcommand`, each of the arguments can receive arbitrary text with a small number of restrictions:

- Braces must be properly balanced because otherwise LaTeX will be unable to determine where the argument ends.
- The `\verb` command, the `verbatim` environment, and related commands or environments are not supported within arguments.

- In an optional argument a closing bracket "]" is allowed only if hidden inside braces (e.g., `\item[{a]}]` is allowed). Without the braces the first] would be misinterpreted as the end of the optional argument.

Deliberately restricting argument contents

The allowed content of arguments can be deliberately further restricted by using the `\newcommand*` variant of the declaration.

`\newcommand*{`*cmd*`}[`*narg*`][`*default*`]{`*command definition*`}`

The starred form works like `\newcommand` but defines a *cmd* that is not, in TeX terms, long. This means that the newly defined command does not accept empty lines or `\par` commands in its argument(s). This restriction can be useful for commands whose arguments are not intended to contain whole paragraphs of text.

Relation to TeX primitives

Commands that have been defined with the low-level TeX primitive `\def` do not accept `\par` in their argument. Thus, they are equivalent to being defined with `\newcommand*`. The low-level TeX equivalent to `\newcommand` is `\long\def`.

Nesting new commands in each other

Sometimes it is necessary to nest command definitions, most commonly in the combination of commands being defined as part of the definition of some new environment. If the inner command (or environment) has arguments there is a problem referring to them. Clearly we cannot use #1, #2, and so on, since this notation already denotes the argument(s) of the outer command or environment. The TeX solution is to double the hash marks; thus, ##1 would refer to the first argument of the inner definition and in case of three nested definitions we would need ####1.

To make this abstract concept a bit clearer, we define a command `\DEFlvec` that (re)defines the `\lvec` command from Example A-1-5 on the preceding page over and over again. As a first argument to `\DEFlvec` we pass the vector name that is being hard-wired into the redefinition of `\lvec`. As the second argument we pass the upper index that will become the default value for the optional argument of `\lvec`. Thus, since the vector name is now part of the definition, `\lvec` has only an optional argument.

```
\newcommand\lvec{}
\newcommand\DEFlvec[2]{\renewcommand\lvec[1][#2]%
         {\ensuremath{#1_1+\cdots + #1_{##1}}}}
\DEFlvec{x}{n}  % initial definition
Default: $\lvec \neq \lvec[k]$ \par
\DEFlvec{y}{i} Now: $\lvec \neq \lvec[k]$
```

Default: $x_1 + \cdots + x_n \neq x_1 + \cdots + x_k$
Now: $y_1 + \cdots + y_i \neq y_1 + \cdots + y_k$

A-1-6

The technique used in the above example is worth studying. Try to visualize the actual definitions being carried out, for example, when the "initial definition" is executed. Also note the need for a top-level definition for `\lvec`: the actual definition is irrelevant but without it we would be unable to "redefine" it inside `\DEF1vec` command.

Special declarations for use in packages and classes

Beside `\newcommand` and `\renewcommand`, which were originally provided as user commands (e.g., for the document preamble), LaTeX offers some extra methods of (re)defining commands that are intended for use in class and package files.

`\providecommand*{`*cmd*`}[`*narg*`][`*default*`]{`*command definition*`}`

This declaration works exactly like `\newcommand` and `\newcommand*`, except that it is ignored if the command to be defined already exists. Such a feature is useful in sources that may get used in several documents, such as bibliography entries. For example, instead of using `\newcommand` in the `@preamble` of BibTeX for logos and other constructs used in the BibTeX entries, you can use `\providecommand` to avoid error messages if such commands are already defined in the document.

`\DeclareRobustCommand*{`*cmd*`}[`*narg*`][`*default*`]{`*command definition*`}`

This command takes the same arguments as `\newcommand` and `\newcommand*` but declares a robust command, even if some code within the *command definition* is fragile. You can use this command to define new robust commands, or to redefine existing commands and make them robust. Information is placed into the transcript file if *cmd* is redefined, so it does not produce an error in this case.

`\CheckCommand*{`*cmd*`}[`*narg*`][`*default*`]{`*command definition*`}`

This command takes the same arguments as `\newcommand` and `\newcommand*` but, rather than defining *cmd*, checks that the current definition of *cmd* is exactly as given by *command definition*. An error is raised if the definitions differ, or if one accepts `\par` in its arguments and the other does not (i.e., was defined using a starred form). This command is useful for checking the state of the system before a package starts altering the definitions of commands. It allows you to check, in particular, that no other package has redefined the same command.

A.1.3 Defining new environments

You can define and redefine an environment with the `\newenvironment` and `\renewenvironment` commands, respectively. You must specify, in each case, which actions should take place when you enter and leave an environment. For an environment called "`myenv`" this is signaled by the commands `\begin{myenv}` and `\end{myenv}` inside your document.

```
\newenvironment{name}[narg]{begdef}{enddef}
\renewenvironment{name}[narg]{begdef}{enddef}
```

As with the \newcommand declaration, the number of arguments is in the range $0 \leq narg \leq 9$. In the case of no parameters, you can omit [0]. Inside the definition part, *begdef*, these parameters are referenced as #1 to #*narg*. If arguments are present, then they are defined when *entering* the environment by specifying them on the command \begin{myenv}, as shown below:

```
\begin{myenv}{arg_1}...{arg_k}
```

Arguments not available in end-tag

When *exiting* an environment with the command \end{myenv} no parameters can be specified. Moreover, the parameters specified with the \begin{myenv} command when entering the environment (see above) are no longer available in the definition part *enddef*, where you define the actions that should take place when leaving the *myenv* environment. This means that it is your responsibility to store information needed at the end of an environment (see the Citation environment defined below).

Technically, a \newenvironment declaration for the environment *myenv* defines a command *myenv* that is called during the \begin{*myenv*} processing and a command \end*myenv* that is executed (besides other things) by \end{*myenv*}. You may find that it is sometimes these commands rather than the environment tags that are used inside packages and classes to define related environments or commands. An example where this might be useful is given on page 468. In other situations, it is not advisable to follow this practice without a thorough understanding of LaTeX's kernel implementation.

Our first example defines an environment of type "Abstract", which is often used to give a short summary of the contents of an article or a book. It starts by typesetting a boldfaced and centered title, followed by the text of the abstract inside a quote environment. The final \par command ensures that any following text starts a new paragraph.

Abstract

This abstract explains the approach used to solve the problems at hand.

Some text following the abstract. Some text following the abstract. And some more.

```
\newenvironment{Abstract}
  {\begin{center}\normalfont\bfseries Abstract%
   \end{center}\begin{quote}}{\end{quote}\par}
\begin{Abstract}
 This abstract explains the approach used
 to solve the problems at hand.
\end{Abstract}
Some text following the abstract. Some text
following the abstract. And some more.
```

A-1-7

Our second example is somewhat more complex. It shows you how a Citation environment can be defined for quoting citations by famous people.

The LaTeX code shown below defines the counter `Citctr`, for numbering the citations, and a box `\Citname`, for storing the name of the person whom we are citing so that we can typeset it at the end of the citation, when the `\end{Citation}` command is encountered (remember that the value of the argument specified on the `\begin{Citation}` command is no longer available at that stage). When entering the environment, we save the value of the argument, typeset in italic, in the box `\Citname` and increment our counter. We then start a `description` environment. This environment will have a single `\item` containing the counter value preceded by the word "Citation". When exiting the `Citation` environment, we twice issue a stretchable horizontal space separated by an allowed—but discouraged—line break. It is important that this space survives if a line break happens before or after it, so `\hspace*` is used. We also throw in a `\quad` of space that ensures a proper separation between the citation and the name if they appear on the same line, but will vanish if a break is taken between them. Then we typeset the contents of the box `\Citname` before leaving the `description` environment. This will put the author's name flush right and the last line of the citation flush left, regardless of whether they end up on separate lines, as you can see in the next example. Without this adjustment the text of the citation would always be fully justified, often with a lot of white space between the words. For a discussion of the counter and box commands used in this example, see Sections A.1.4 and A.2.

Citation 1 Man is the measure of all things. *Protagoras*

This is some regular text in between two Citation environments.

Citation 2 On mourra seul. *Blaise Pascal*

More regular text ...

Citation 3 Necessity is the plea for every infringement of human freedom.
William Pitt

A-1-8

```
\newcounter{Citctr} \newsavebox{\Citname}
\newenvironment{Citation}[1]
   {\sbox\Citname{\emph{#1}}%
    \stepcounter{Citctr}\begin{description}
      \item[Citation \arabic{Citctr}]}
   {\hspace*{\fill}\nolinebreak[1]%
    \quad\hspace*{\fill}%
%% \finalhyphendemerits=0 %% see text below
    \usebox{\Citname}\end{description}}
\begin{Citation}{Protagoras} Man is the
 measure of all things.     \end{Citation}
This is some regular text in between two
Citation environments.
\begin{Citation}{Blaise Pascal}
 On mourra seul.            \end{Citation}
More regular text \ldots
\begin{Citation}{William Pitt} Necessity
 is the plea for every infringement of
 human freedom.             \end{Citation}
```

Surprisingly, the name in the last citation is typeset on a line of its own, even though there is clearly enough space to place it alongside with the citation. The reason is that TeX's paragraph-breaking algorithm prefers solutions that do not have the second-to-last line ending in a hyphen and therefore selects a three-line paragraph breaking at the `\nolinebreak`.

A hyphen on the second-to-last line of a paragraph

There are two ways to correct this behavior. First, we can discourage breaking at this point by using an optional argument of `[3]` instead of `[1]`, which would work in that particular example but may not work always. Second, we can tell TeX's algorithm not to take that hyphen into account by setting the low-level TeX integer parameter `\finalhyphendemerits` to zero. This requires a somewhat unusual syntax, as shown in the example code above (though commented out there to display the behavior without it).

As with `\newcommand` one can make the first argument of an environment optional:

`\newenvironment{`*name*`}[`*narg*`][`*default*`]{`*begdef*`}{`*enddef*`}`

The *default* value for the optional argument is given between the second pair of square brackets. Inside the *begdef* part, which is executed when the environment *name* is entered, the optional argument can be accessed with `#1`. The mandatory arguments (when present) are addressed as `#2` to `#`*narg*. When the *name* environment is used without an optional parameter, `#1` will contain the string specified as *default*.

As an example, we reimplement the `altDescription` environment from Example 3-3-27 on page 149, this time with an optional argument instead of a mandatory argument specifying the width of the indentation. Another difference from the earlier definition is that the list labels will be placed flush right if possible (by placing `\hfil` at the left in `\makelabel`). When used without an optional argument the indentation will be `1em` (i.e., a `\quad`). By specifying the widest entry as an optional argument, you will make sure that the description parts of all your entries line up nicely.

The example first shows the (default) behavior of the `altDescription` list, then displays what it looks like when using the optional argument.

First This is a short term with text that wraps.

Long term This is a long term.

Even longer term A very long term.

First This is a short term with text that wraps.

Long term This is a long term.

Even longer term A very long term.

```
\usepackage{calc}
\newenvironment{altDescription}[1][\quad]%
  {\begin{list}{}{%
    \renewcommand\makelabel[1]{\hfil\textsf{##1}}%
    \settowidth\labelwidth{\makelabel{#1}}%
    \setlength\leftmargin{\labelwidth+\labelsep}}}
  {\end{list}}
\begin{altDescription}
\item[First] This is a short term with text that wraps.
\item[Long term] This is a long term.
\item[Even longer term] A very long term.
\end{altDescription}
\begin{altDescription}[Even longer term]
\item[First] This is a short term with text that wraps.
\item[Long term] This is a long term.
\item[Even longer term] A very long term.
\end{altDescription}
```

A-1-9

A.1.4 Defining and changing counters

Every number internally generated by LaTeX has a *counter* (register) associated with it. The name of the counter is usually identical to the name of the environment or the command that generates the number except that it does not start with \. The following is the list of all counters used in LaTeX's standard document classes:

`part`	`paragraph`	`figure`	`enumi`
`chapter`	`subparagraph`	`table`	`enumii`
`section`	`page`	`footnote`	`enumiii`
`subsection`	`equation`	`mpfootnote`	`enumiv`
`subsubsection`			

An environment declared by `\newtheorem` can also have a counter with the same name associated with it, unless the optional argument indicates that it is to be numbered together with another environment.

The value of a counter is a single integer. Several counters can be combined into a number, as is usually the case for numbering section headings. For example, in the book or report classes, 7.4.5 identifies the fifth subsection of the fourth section in the seventh chapter.

Below we describe all the basic LaTeX commands that define counters and modify or display their values. These commands are much more powerful if used in conjunction with the calc package, which is discussed in Section A.3.1.

`\newcounter{`*newctr*`}[`*oldctr*`]`

This command globally defines a new counter, *newctr*, and initializes it to zero. If a counter with the name *newctr* is already defined, an error message is printed. When you specify the name of another counter as the optional argument, *oldctr*, then the newly defined *newctr* is reset when the counter *oldctr* is incremented with the `\stepcounter` or `\refstepcounter` command. It also defines the command `\the`*newctr* to expand to `\arabic{`*newctr*`}`.

`\@addtoreset{`*reset-ctr*`}{`*ctr*`}` `\@removefromreset{`*reset-ctr*`}{`*ctr*`}`

The operation that defines that one counter is reset whenever another counter is stepped is also available as the kernel command `\@addtoreset`.[1] Unfortunately, the opposite declaration is not available in the kernel, but only when loading the package remreset. If this small package is loaded, then counters can be unraveled if necessary. For example, the report class defines that the `footnote` counter is to be reset whenever a new chapter starts. If you want your footnotes nevertheless

Warning: `\@removefromreset` *needs a package!*

[1] See also the `\numberwithin` declaration provided by the amsmath package. It is discussed in Section 8.2.14 on page 485.

to be numbered sequentially throughout a document, then specifying

```
\usepackage{remreset}
\makeatletter \@removefromreset{footnote}{chapter} \makeatother
```

in the preamble, or the equivalent code[1] in a package or class, will do the job.

`\setcounter{`*ctr*`}{`*val*`}` `\addtocounter{`*ctr*`}{`*val*`}`

With `\setcounter` the value of counter *ctr* is globally set equal to the value *val*. With `\addtocounter` it is globally incremented by *val*.

`\stepcounter{`*ctr*`}` `\refstepcounter{`*ctr*`}`

Both commands globally increment the counter *ctr* and reset all subsidiary counters—that is, those declared with the optional argument *oldctr* on the `\newcounter` command or with the first argument of `\@addtoreset`. The `\refstepcounter` command additionally defines the current `\ref` value to be the text generated by the command `\the`*ctr*. Note that whereas stepping a counter is a global operation, setting the current `\ref` value is done locally and thus is only valid inside the current group. As a result the next example does not produce the desired result but instead picks up the section number. The correct solution would be to move `\refstepcounter` before the `\textbf` command.

5 A Failure

Exercise 5.a: A test.

Exercise 5.b: Another test.

Referencing exercises: 5 and 5.

```
\newcounter{ex} \renewcommand\theex{\thesection.\alph{ex}}
\newenvironment{EX}{\begin{flushleft}%
    \textbf{\refstepcounter{ex}Exercise~\theex:}}
   {\end{flushleft}}
\setcounter{section}{4} % for testing
\section{A Failure}
\begin{EX} \label{A} A test.       \end{EX}
\begin{EX} \label{B} Another test. \end{EX}
Referencing exercises: \ref{A} and \ref{B}.
```

A-1-10

`\value{`*ctr*`}` `\arabic{`*ctr*`}` `\roman{`*ctr*`}` `\Roman{`*ctr*`}`
`\alph{`*ctr*`}` `\Alph{`*ctr*`}` `\fnsymbol{`*ctr*`}`

The `\value` command produces the current value of a counter to be used in places where LaTeX expects to see a number, such as in the *val* argument of the command `\setcounter` or `\addtocounter` or when comparing numbers using the `\ifthenelse` command from the ifthen package. However, the command cannot be used to typeset the value of the counter! For that purpose a set of presentation commands are available, all of which take a counter name as argument.

[1]In that case use `\RequirePackage` and *omit* `\makeatletter` and `\makeatother`!

With `\arabic` the counter value is represented as an Arabic numeral. With `\roman` and `\Roman` lowercase and uppercase Roman numerals are produced, respectively.

The remaining commands can be used only if the counter value is within a certain range. The `\alph` command displays the value as a lowercase letter: a, b, c, ..., z. Thus, the value should lie in the range 1, ..., 26; otherwise, an error is signaled. The `\Alph` command is similar but produces uppercase letters. Finally, `\fnsymbol` represents the counter value as a traditional footnote symbol (e.g., $*$, †). In that case the value must not be greater than 9, unless an extension package, like footmisc, is used. The next example shows all of these commands in action.

Counter presentations with restricted ranges

8, viii, VIII, h, H, ††
Anno Domini MCMXCIV

```
\newcounter{exa}\setcounter{exa}{8}
\arabic{exa}, \roman{exa}, \Roman{exa}, \alph{exa},
\Alph{exa}, \fnsymbol{exa} \par
\setcounter{exa}{1994} Anno Domini \scshape{\roman{exa}}
```

A-1-11

`\the`⟨*ctr*⟩

A shorthand to produce the default visual representation for a counter *ctr* is provided by the command `\the`⟨*ctr*⟩ (e.g., `\thesection` for the `section` counter). As mentioned earlier this command is initialized by the `\newcounter` declaration to produce `\arabic{`*ctr*`}`. However, in LaTeX such a visual representation often involves more than a single number. For example, with sectioning counters one usually displays the value of the current section as well as the value of the current subsection, and so on. For this reason `\the`⟨*ctr*⟩ is typically (re)defined to produce a more complex representation. This practice becomes even more important when you consider that `\refstepcounter` not only increments a certain counter and resets lower-level counters but also defines the "current" label (as picked up by `\label`) to be the result of `\the`⟨*ctr*⟩ for the counter being stepped.

As an example, inside the standard article class, we find definitions for sectioning counters equivalent to the following:

```
\newcounter{part}          \newcounter{subsection}[section]
\newcounter{section}       \newcounter{subsubsection}[subsection]
\renewcommand\thepart         {\Roman{part}}
\renewcommand\thesection      {\arabic{section}}
\renewcommand\thesubsection   {\thesection.\arabic{subsection}}
\renewcommand\thesubsubsection{\thesubsection.\arabic{subsubsection}}
```

You see how lower-level counters are reset when upper-level counters are stepped, as well as how the representation of the counters (the `\the...` commands) are constructed from the current counter and the counters at a higher level. Note how the `part` counter does not influence any of the lower levels.

As another example, we look at Table 3.6 on page 130, which shows the structure of the enumeration list counters. In fact, these counters are defined inside the file `latex.ltx`, which contains the kernel code for LaTeX. Only the representation, prefix, and label field commands are defined in the standard class files as follows:

```
\renewcommand\theenumi  {\arabic{enumi}}       \renewcommand\theenumii{\alph{enumii}}
\renewcommand\theenumiii{\roman{enumiii}}       \renewcommand\theenumiv{\Alph{enumiv}}

                                                \renewcommand\p@enumii{\theenumi}
\renewcommand\p@enumiii{\theenumi(\theenumii)} \renewcommand\p@enumiv{\p@enumiii\theenumiii}

\newcommand\labelenumi  {\theenumi.}            \newcommand\labelenumii{(\theenumii)}
\newcommand\labelenumiii{\theenumiii.}          \newcommand\labelenumiv{\theenumiv.}
```

Finally, we show how the standard classes handle the `equation` counter. Like the enumeration counters, this counter is declared inside `latex.ltx`. In the article class the counter is never reset:

```
\renewcommand\theequation{\arabic{equation}}
```

In the report and book classes the equation number is reset for each chapter with the `\@addtoreset` command:

```
\@addtoreset{equation}{chapter}
\renewcommand\theequation{\thechapter.\arabic{equation}}
```

Also, the representation differs in both cases.[1]

A.1.5 Defining and changing space parameters

In (LA)TeX two kinds of space parameters (lengths) exist: "rigid" lengths (called `<dimen>` in *The TeXbook* [82]), which are fixed, and "rubber" lengths (called `<skip>` in *The TeXbook*), which have a natural length and a degree of positive and negative elasticity. New lengths in LaTeX are allocated as type `<skip>`, so that you always have the choice of initializing them as rigid or rubber lengths (by specifying `plus` and `minus` parts). On the other hand, all standard lengths in LaTeX are of type rigid, unless specifically declared in Appendix C of the *LaTeX Manual* to be rubber. Here we discuss the commands provided by LaTeX for dealing with lengths.

`\newlength{`*cmd*`}`

The declaration `\newlength` allocates a new (rubber) length register and associates the command name *cmd* with it. If a command *cmd* already exists, you will get an error message. The new length is preset to zero. Just like with `\newcommand` you will find that the braces around *cmd* are often omitted in actual code since the argument must consist of a single command name.

[1] The actual definition is somewhat more complex, since some low-level code is used to suppress the chapter number if it is zero.

sp	Scaled point (65536sp = 1pt) TeX's smallest unit	
pt	Point = $\frac{1}{72.27}$ in = 0.351mm	ı
bp	Big point = $\frac{1}{72}$ in = 0.353mm, also known as PostScript point	ı
dd	Didôt point = $\frac{1}{72}$ of a French inch, = 0.376mm	ı
mm	Millimeter = 2.845pt	⊔
pc	Pica = 12pt = 4.218mm	⊔
cc	Cicero = 12dd = 4.531mm	⊔
cm	Centimeter = 10mm = 2.371pc	⊔
in	Inch = 25.4mm = 72.27pt = 6.022pc	
ex	Height of a small "x" in the current font (approximately)	⊔
em	Width of capital "M" in current font (approximately)	⊔
mu	Math unit (18mu = 1em) for positioning in math mode	ı

Table A.1: LaTeX's units of length

`\setlength{`*cmd*`}{`*length*`}`   `\addtolength{`*cmd*`}{`*length*`}`

This sets the value of the length command *cmd* equal to the length *length* or, in case of `\addtolength`, adds the specified amount to the existing value. In the examples below, the TeX command `\the` is used to typeset the actual contents of the length variable. It requires the register command name *without* braces!

Mylen = 28.45274pt
A-1-12 Mylen = 28.45274pt plus 4.0pt minus 2.0pt

```
\newlength\Mylen
\setlength  \Mylen{10mm}  Mylen = \the\Mylen
\addtolength\Mylen{0pt plus 4pt minus 2pt}
\par Mylen = \the\Mylen
```

Lengths can be specified in various units, as shown in Table A.1. Notice the difference between the typographic point (pt), which is normally used in TeX, and the (big) point used by PostScript, for example. Thus, when reserving space for an EPS picture you need to specify the bounding box dimension in bp to get the correct space.

`\settowidth{`*cmd*`}{`*text*`}`
`\settoheight{`*cmd*`}{`*text*`}`   `\settodepth{`*cmd*`}{`*text*`}`

Instead of specifying a length value explicitly, three commands are available that allow you to measure a given text and assign the result. With `\settowidth` the value of the length command *cmd* is set equal to the natural width of the typeset version of *text*. This command is very useful for defining lengths that vary with the string contents or the type size. The other two commands work similarly but

`\hspace{`*len*`}`	Horizontal space of width *len* that can be a rigid or a rubber length
`\enspace`	Horizontal space equal to half a quad
`\quad`	Horizontal space equal to the em value of the font
`\qquad`	Twice a `\quad`
`\hfill`	Horizontal rubber space that can stretch between 0 and ∞
`\hrulefill`	Similar to `\hfill`, but draws a solid horizontal line
`\dotfill`	Similar to `\hfill`, but draws a dotted line

Table A.2: Predefined horizontal spaces

measure the height and the depth rather than the width of the typeset *text*.

width = 48.03pt
height = 6.7799pt
depth = 2.16492pt
Use larger font and recalculate:
width = 57.63602pt

```
\newlength\Mylen    \raggedright% to make example nicer
\settowidth \Mylen{Typography} width  = \the\Mylen \\
\settoheight\Mylen{Typography} height = \the\Mylen \\
\settodepth \Mylen{Typography} depth  = \the\Mylen \par
Use larger font and recalculate: \\
\settowidth\Mylen{\large Typography} width = \the\Mylen
```

A-1-13

`\fill`   `\stretch{`*dec-num*`}`

These two rubber lengths are intended to be used in the argument of `\vspace` and similar commands. The `\fill` rubber length is preset with a natural length of zero but can stretch to any positive value. Do not change its value! It is used in various places in the kernel and a change would produce strange effects.

An often more useful rubber length is provided by the `\stretch` command—in fact, `\fill` is equivalent to `\stretch{1}`. More generally, `\stretch{`*dec-num*`}` has a stretchability of *dec-num* times `\fill`. It can be used to fine-tune the positioning of text horizontally or vertically—for instance, to provide spaces that have a certain relation to each other. Example A-1-15 demonstrates its application.

Horizontal space

Table A.2 shows horizontal space commands known to LaTeX. A flexible horizontal space of any desired width is produced by the `\hspace` command. The command `\hspace*` is the same as `\hspace`, but the space is never removed—not even at a line boundary.

A space in front of or following an `\hspace` or `\hspace*` command is significant, as the following example shows:

This is a   0.5 in wide space.
This is a   0.5 in wide space.
This is a   0.5 in wide space.

```
\par This is a\hspace{0.5in}0.5~in wide space.
\par This is a \hspace{0.5in}0.5~in wide space.
\par This is a \hspace{0.5in} 0.5~in wide space.
```

A-1-14

The next example shows how rubber lengths can be used to fine-tune the positioning of information on a line. Note that the `\hfill` command is, in fact,

`\smallskip`	Vertical skip of `\smallskipamount` (default about one quarter of `\baselineskip`)
`\medskip`	Vertical skip of `\medskipamount` (default about one half of `\baselineskip`)
`\bigskip`	Vertical skip of `\bigskipamount` (default about one `\baselineskip`)
`\vfill`	Vertical rubber length that can stretch between 0 and ∞

Table A.3: Predefined vertical spaces

an abbreviation for `\hspace{\fill}`. To save typing, we also defined a command with an optional argument, `\HS`, which behaves like `\hfill` when used without an argument, but can be made less or more flexible than that command by specifying the stretchability (a value of 1 has the same effect as `\hfill`).

left right
left $\boxed{\frac{2}{5}}$ right
left middle right
left middle ______________ right
left .... right
left .... right
left .... right
A-1-15 left .... right

```
\newcommand{\HS}[1][1.]{\hspace{\stretch{#1}}}
\begin{center}
left \hfill                              right\\
left \HS[2]\fbox{$\frac{2}{5}$}\HS[5]    right\\
left \HS        middle \hfill            right\\
left \hrulefill\ middle \hrulefill\      right\\
left \dotfill\                           right\\
left \dotfill\ \HS[.5] \dotfill\         right\\
left \dotfill\ \HS     \dotfill\         right\\
left \dotfill\ \HS[2.] \dotfill\         right
\end{center}
```

Vertical space

A vertical space is produced with the `\vspace` command, which works similarly to `\hspace`. In particular, a `\vspace*` command will generate vertical space that will never be eliminated, even when it falls on a page break where a `\vspace` command will be ignored at this point. Table A.3 shows vertical space commands known to LaTeX that are common to all standard classes.

LaTeX users are often confused about the behavior of the `\vspace` command. When used inside a paragraph, the vertical space is added after the end of the line with `\vspace`; between paragraphs it behaves as you would expect.

The use of a `\vspace` command inside

a paragraph is considered somewhat odd. It could perhaps be used with a negative space value to get rid of redundant space.

Between paragraphs, adjusting the spacing is somewhat more useful, and it allows control of the white space before and after dis-
A-1-16 played material.

```
The \vspace{3mm}use of a \verb!\vspace! command
inside a paragraph is considered somewhat odd.
It could perhaps be used with a negative space
value to get rid of redundant space.

\vspace{\baselineskip}

Between paragraphs, adjusting the spacing is
somewhat more useful, and it allows control
of the white space before and after displayed
material.
```

Stretchable space as introduced on page 856 can also be used for vertical material. The `\vfill` command is, in fact, an abbreviation for a blank line followed by `\vspace{\fill}`. More generally, you can use the `\stretch` command in combination with `\vspace` to control the layout of a complete page. This could be useful for designing a title page: if the title should be placed one third of the way down the page, one simply has to place `\vspace*{\stretch{1}}` before it and `\vspace*{\stretch{2}}` after it.

Geoffrey Chaucer
The Canterbury Tales

LONDON 1400

```
\newcommand\HRule{\noindent\rule{\linewidth}{1.5pt}}
\begin{titlepage}
  \vspace*{\stretch{1}}
  \HRule
  \begin{flushright}
    \LARGE   Geoffrey Chaucer  \\
             The Canterbury Tales
  \end{flushright}
  \HRule
  \vspace*{\stretch{2}}
  \begin{center}
    \textsc{London 1400}
  \end{center}
\end{titlepage}
```

A-1-17

`\addvspace{`*space*`}`

Use with care if at all

While LaTeX's user command `\vspace` unconditionally adds a vertical space (which is removed only at page boundaries, while its starred form even suppresses this action), there exists another command for adding vertical space that is often used in the kernel and in some package files. The `\addvspace` command has somewhat different semantics, and although it appears to be a user-level command judging from its name, in fact it is not.

In contrast to `\vspace` the command `\addvspace` is allowed only in vertical mode (i.e., between paragraphs). If used in horizontal mode, it issues the famous "`Something's wrong-perhaps a missing \item`" error, which most LaTeX users know and love. Most of the time this error has nothing to do with a missing or misplaced `\item` but simply signals a misplaced `\addvspace` command. But it shows some of the history of this command: originally, it was developed and used solely for spacing items in `list` environments.

The other important semantic difference between `\vspace` and `\addvspace` is that the latter adds a space whose size depends on any directly preceding space. The precise rules are inherited from LaTeX 2.09 and show some strange discontinuities that nobody these days seems to be able to explain fully, though for backward compatibility the command is retained in this form. If s is the space to be added by `\addvspace` and ℓ is the size of the vertical space (if any) before the current point, then the following rules apply:

If	$s < 0\text{pt} < \ell$	do	backup by s
elseif	$\ell = 0\text{pt}$	do	add an additional space of s
		else	make a space of $\max(\ell, s)$ out of the two

If we ignore for the moment the special cases in the first two lines of the rules, then the idea behind `\addvspace` can be described as follows: if we have two vertically oriented constructs, such as a list and a heading, and both want to surround themselves with some vertical spacing before and after, it is probably not a good idea if both such spaces are applied if the objects directly follow each other. In that case using the maximum of both spaces is usually a better solution. This is why lists, headings, and other typeset elements use `\addvspace` rather than `\vspace`.

This has some rather surprising effects. If you have two such display objects following each other, then only the maximum of the space surrounding them is used. But if you try to enlarge that space slightly, such as by placing `\vspace{4pt}` between them, then suddenly the space will be far larger. This result occurs because in a sequence like

Surprising space size changes

```
\addvspace{10pt} \vspace{4pt} \addvspace{8pt}
```

the second `\addvspace` will be unable to see the first and will add all of its space (with the result that the total space is 22 pt); without the `\vspace` in the middle you would get 10 pt total. The `\vspace` does not interact with the following `\addvspace` because it actually generates a space of 4 pt followed by a space of 0 pt, so that the second rule applies.

If you notice that your space got too large and you reduce your correction to, say, `\vspace{2pt}`, nothing will change substantially (you still get 20 pt). Even more surprisingly, if you try to make the original space smaller by using, say, `\vspace{-3pt}`, you will end up with 15 pt total space—still more than before.

To actually get a space of 7 pt in that place, you would need to back up by 11 pt. Unfortunately, there is no way to determine the size of the necessary space other than by experimenting or looking into the definitions of the objects above and below, to find out what `\addvspace` values are used at a given point.

The same problem arises if some other invisible object separates two consecutive `\addvspace` commands. For example, a color-changing command or a `\label` will effectively hide a previous `\addvspace`, with the result that suddenly not the maximum, but the sum of both spaces, appears.

`\addpenalty{`*penalty*`}`

Although `\addpenalty` is not a spacing command it is described here because it is intended to work together with `\addvspace`. A penalty is TeX's way of assigning a "badness" to break points. A high penalty means that this is a bad place to break, while a negative penalty indicates to TeX that this is a rather good place to start a new line or a new page. Details of this mechanism can be found in Chapters 14 and 15 of [82].

The `\addpenalty` command requires a TeX penalty value as an argument (useful values are between -10000 and 10000). For example, `\@startsection` discussed in Chapter 2 uses `\addpenalty` to make the space before a heading become a good place to break (default value -300). If `\addpenalty` and `\addvspace` are mixed, then this has two effects:

- LaTeX will still use the maximum of the spaces even if `\addpenalty` appears between two `\addvspace` commands.

- LaTeX moves the potential break "visually" to the beginning of the white space, even if there is an `\addvspace` before the `\addpenalty`.

The second feature is important to avoid white space remaining at the bottom of pages. See page 937 for a discussion of how this is achieved.

A.2 Page markup—Boxes and rules

The theory of composing pages out of boxes lies at the very heart of TeX, and several LaTeX constructs are available to take advantage of this method of composition. A *box* is a rectangular object with a height, depth, and width. Its contents can be arbitrarily complex, involving other boxes, characters, spaces, and so forth. Once built it is used by LaTeX as a single, fixed object that behaves much like a (potentially huge) character. A box cannot be split and broken across lines or pages. Boxes can be moved up, down, left, and right. LaTeX has three types of boxes:

LR (left-right) The contents of this box are typeset from left to right. Line breaking is impossible and commands like `\\` and `\newline` are ignored or produce error messages.

Par (paragraphs) This kind of box can contain several lines, which will be typeset in paragraph mode just like normal text. Paragraphs are put one on top of the other. Their widths are controlled by a user-specified value.

Rule This (thin or thick) line is often used to separate various logical elements on the output page, such as table rows and columns, and running titles and the main text.

LaTeX's boxes all start a paragraph (just like characters) if used in vertical mode, while TeX's primitive box commands (e.g., `\hbox`) behave differently depending on where they are used. There are a number of reasons to avoid using the TeX primitives directly; see the discussion in Section A.2.5. The situation with rules is slightly different; we therefore will discuss TeX's primitive rule commands below.

A.2.1 LR boxes

`\mbox{`*text*`}` `\fbox{`*text*`}`
`\makebox[`*width*`] [`*pos*`] {`*text*`}` `\framebox[`*width*`] [`*pos*`] {`*text*`}`

The first line considers the *text* inside the curly braces as a box, without or with a frame drawn around it. For example, `\fbox{some words}` gives $\boxed{\text{some words}}$. The two commands on the second line are a generalization of these commands. They allow the user to specify the width of the box and the positioning of the text inside.

some words	`\makebox[5cm]{some words} \par`
some words	`\framebox[5cm][r]{some words}`

A-2-1

In addition to centering the text with the positional argument `[c]` (the default), you can position the text flush left (`[l]`) or flush right (`[r]`). There is also an `[s]` specifier that will stretch your *text* from the left margin to the right margin of the box provided it contains some stretchable space (e.g., some `\hspace` or the predefined spaces given in Table A.2 on page 856). Interword spaces are also stretchable (and shrinkable to a certain extent), as explained on page 428. The appearance of frameboxes can be controlled by two style parameters:

`\fboxrule` The width of the lines for the box produced with the command `\fbox` or `\framebox`. The default value in all standard classes is 0.4 pt.

`\fboxsep` The space left between the edge of the box and its contents by `\fbox` or `\framebox`. The default value in all standard classes is 3 pt.

Any changes to these parameters obey the normal scoping rules and affect all frameboxes within the scope. The change to `\fboxsep` in the next example, for instance, applies only to the second box.

A-2-2

Boxed Text   Boxed Text   Boxed Text

```
\fbox{Boxed Text} \hfill
\setlength\fboxrule{2pt}%
{\setlength\fboxsep{2mm}\fbox{Boxed Text}}
\hfill \fbox{Boxed Text}
```

The box commands with arguments for specifying the dimensions of the box allow you to make use of four special length parameters: `\width`, `\height`, `\depth`, and `\totalheight`. They specify the natural size of the *text*, where `\totalheight` is the sum of `\height` and `\depth`.

A-2-3

A few words of advice

A few words of advice

A few words of advice

```
\usepackage{calc}
\framebox{ A few words of advice }              \par
\framebox[\width + 8mm][s]{ A few words of advice }
\par  \framebox[1.5\width]{ A few words of advice }
```

Zero-width boxes are very handy if you want to put a marker on the page (e.g., for placement of figures) or to allow text to be put into the margins. The principle of operation is shown below, where a zero-width box is used to tag text, without influencing the centering. Note that the optional parameter `[l]` (`[r]`) makes the material stick out to the right (left).

A-2-4

A sentence.^{123}

Some more text in the middle.

^{321}A sentence.

```
\centering
A sentence.\makebox[0pt][l]{\textsuperscript{123}}\\
Some more text in the middle.                      \\
\makebox[0cm][r]{\textsuperscript{321}}A sentence.
```

⟺As seen in the margin of the current line, boxes with a vanishing width can be used to make text stick out into the margin. This effect was produced by beginning the current paragraph in the following way:

```
\noindent\makebox[0cm][r]{$\Longleftrightarrow$}%
As seen in the margin ...
```

An interesting possibility is to raise or lower boxes. This can be achieved by the very powerful `\raisebox` command, which has two mandatory arguments and two optional arguments:

`\raisebox{`*lift*`}[`*height*`][`*depth*`]{`*contents*`}`

To raise or lower the box produced from the *contents*, one specifies the amount of *lift* as a dimension, with negative values lowering the box. As with other boxes, one can make use of the special commands `\height`, `\depth`, `\totalheight`, or even `\width` to refer to the natural dimensions of the box produced from *contents*. This is used in the next example to raise the word "upward" so that the descender of the "p" aligns with the baseline and to lower the word "downward" so that it is placed completely below the baseline.

x111x upward x222x downward x333x

```
x111x \raisebox{\depth}{upward} x222x
  \raisebox{-\height}{downward} x333x
```

A-2-5

Normally, LaTeX takes the added height and depth into account when calculating the distance between the lines, so that a raised or lowered box can result in spreading lines apart. This can be manipulated by specifying a *height* and a *depth* that the user wants LaTeX to actually use when placing its material on the page. The second pair of lines below shows that LaTeX does not realize that text has been moved upward and downward; thus, it composes the lines as though all the text was on the baseline.

x111x downward x222x
x333x upward x444x

x111x downward x222x
x333x upward x444x

```
\begin{flushleft}
x111x \raisebox{-1ex}{downward} x222x      \\
x333x \raisebox{1ex}{upward} x444x         \\[4mm]
x111x \raisebox{-1ex}[0cm][0cm]{downward} x222x\\
x333x \raisebox{1ex}[0cm]{upward} x444x
\end{flushleft}
```

A-2-6

A somewhat more useful application is discussed in Section 5.7 on page 272, which addresses the subject of columns spanning multiple rows in `tabular` material.

A.2.2 Paragraph boxes

Paragraph boxes are constructed using the `\parbox` command or `minipage` environment. The *text* material is typeset in paragraph mode inside a box of width

width. The vertical positioning of the box with respect to the text baseline is controlled by the one-letter optional parameter *pos* ([c], [t], or [b]).

\parbox[*pos*]{*width*}{*text*}

\begin{minipage}[*pos*]{*width*}
text
\end{minipage}

The center position is the default, as shown in the next example. Note that LaTeX might produce wide interword spaces if justification is requested (default) and the measure is incredibly small.

This is the contents of the left-most parbox. CURRENT LINE This is the right-most parbox. Note that the typeset text looks sloppy because LaTeX cannot nicely balance the material in these narrow columns.

```
\parbox{.3\linewidth}{This is
  the contents of the left-most
  parbox.}
\hfill CURRENT LINE \hfill
\parbox{.3\linewidth}{This is
  the right-most parbox.
  Note that the typeset text
  looks sloppy because \LaTeX{}
  cannot nicely balance the
  material in these narrow
  columns.}
```

A-2-7

The minipage environment is very useful for the placement of material on the page. In effect, it is a complete miniversion of a page and can contain its own footnotes, paragraphs, and array, tabular, multicols, and other environments. Note, however, that it cannot contain floats or \marginpar commands, but it can appear inside figure or table environments, where it is often used for constructing a pleasing layout of the material inside the float. A simple example of a minipage environment at work is given below. The baseline is shown with an en dash generated by the command \HR. Note the use of the *pos* placement parameter ([c], [t], or [b]) on the three minipage environments.

```
A A A
A A A  B B B B
A A A  B B B B
A A A  B B B B
_A A A _B B B B_C C C C_
        B B B B  C C C
        B B B B
        B B B B
```

```
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR
\begin{minipage}[b]{12mm}
      A A A A A A A A A A A A A A A
\end{minipage}\HR
\begin{minipage}[c]{12mm}
      B B B B B B B B B B B B B B B B B B B B B B B B
\end{minipage}\HR
\begin{minipage}[t]{12mm}
      C C C C C C C
\end{minipage}\HR
```

A-2-8

If you desire more complicated alignments, then you might have to stack the different minipage environments. Compare the behavior of the next examples.

Below, we try to align the two leftmost blocks at their top and align the resulting block at the bottom with a third block by adding another level of minipages.

```
                  C C C C
_A  A  A xx B B B B _C C C  _
 A  A  A    B B B B
 A  A  A    B B B B
 A  A  A    B B B B
 A A A      B B B B
            B B B B
```

```
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR\begin{minipage}[b]{30mm}
   \begin{minipage}[t]{12mm}
        A A A A A A A A A A A A A A A
   \end{minipage} xx  \begin{minipage}[t]{12mm}
      B B B B B B B B B B B B B B B B B B B B B B B B
   \end{minipage}
\end{minipage}\HR
\begin{minipage}[b]{12mm}  C C C C C C C \end{minipage}\HR
```

A-2-9

However, we do not get the expected result. Instead, the two top-aligned minipages inside the bottom-aligned minipage form a paragraph with a single line (the minipages are considered to be large units in the line containing xx). Thus, the bottom line of the outer minipage is still the one containing the xx characters. To prevent this we need to add some invisible space after the paragraph, as shown next.

```
 A  A  A xx B B B B
 A  A  A    B B B B
 A  A  A    B B B B
 A  A  A    B B B B
 A A A      B B B B  C C C C
_           B B B B _C C C  _
```

```
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR\begin{minipage}[b]{30mm}
   \begin{minipage}[t]{12mm}
        A A A A A A A A A A A A A A A
   \end{minipage} xx  \begin{minipage}[t]{12mm}
      B B B B B B B B B B B B B B B B B B B B B B B B
   \end{minipage}
   \par\vspace{0mm}
\end{minipage}\HR
\begin{minipage}[b]{12mm}  C C C C C C C \end{minipage}\HR
```

A-2-10

In the case below, the two rightmost environments are aligned at their top inside another enclosing environment, which is aligned at its bottom with the first one. If you compare it with the previous example, then you see that you obtain a quite different result, although the sequence of alignment parameters is the same. Only the stacking order of the minipage environments is different.

```
          B B B B xx C C C C
 A  A  A B B B B     C C C
 A  A  A B B B B
 A  A  A B B B B
 A  A  A B B B B
_A A A  _B B B B            _
```

```
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR\begin{minipage}[b]{12mm}
   A A A A A A A A A A A A A A A \end{minipage}\HR
\begin{minipage}[b]{30mm}  \begin{minipage}[t]{12mm}
      B B B B B B B B B B B B B B B B B B B B B B B B
 \end{minipage} xx
 \begin{minipage}[t]{12mm}  C C C C C C C  \end{minipage}
 \par\vspace{0mm}
\end{minipage}\HR
```

A-2-11

Again, we had to add some vertical space to achieve alignment. This does not, however, always produce the desired result. If, for instance, a letter with a descender appears in the last line of the stacked minipage, as in the example below, then the alignment of the baselines is not perfect.

```
        B B B B xx C C C C
        B B B B    C C C
A A A   B B B B
A A A   B B B B
A A A   B B B B
A A A   B B B B
_A A A  _gg jj         _
```

A-2-12

```
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR\begin{minipage}[b]{12mm}
      A A A A A A A A A A A A A A A \end{minipage}\HR
\begin{minipage}[b]{30mm}  \begin{minipage}[t]{12mm}
     B B B B B B B B B B B B B B B B B B B B B B B B gg jj
  \end{minipage} xx
  \begin{minipage}[t]{12mm}  C C C C C C C  \end{minipage}
  \par\vspace{0mm}
\end{minipage}\HR
```

To correct this problem, you have to add (negative) vertical space that compensates for the depth of the letters.

Perhaps the easiest way (albeit the most dangerous) is to use the TEX primitive `\prevdepth`. This dimension register can be used only in vertical mode (i.e., after a paragraph has ended), and contains the depth of the previous line. In the next example this primitive is used to back up by this amount, thereby pretending that the bottom of the box is located at the baseline of the last line.

When using `\prevdepth` in this way one has to be careful. As already mentioned, it gives an error if used outside vertical mode. Furthermore, TEX overloads this primitive by setting it to −1000pt at the beginning of a vertical box and after a horizontal rule.[1] Thus, using `\vspace*` instead of `\vspace` in the example would give a nasty surprise, because `\vspace*` actually puts in an invisible rule to ensure that the space will survive at a page break. As a result the value of `\prevdepth` inside would be −1000pt and we would effectively be adding a space of 1000 points at the bottom of the box.

Surprising effects of `\prevdepth`

```
        B B B B xx C C C C
        B B B B    C C C
A A A   B B B B
A A A   B B B B
A A A   B B B B
A A A   B B B B
_A A A  _gg jj         _
```

A-2-13

```
\newcommand\HR{\rule{.5em}{0.4pt}}
\HR\begin{minipage}[b]{12mm}
      A A A A A A A A A A A A A A A \end{minipage}\HR
\begin{minipage}[b]{30mm}  \begin{minipage}[t]{12mm}
     B B B B B B B B B B B B B B B B B B B B B B B B gg jj
     \par\vspace{-\prevdepth}
  \end{minipage} xx
  \begin{minipage}[t]{12mm}  C C C C C C C  \end{minipage}
  \par\vspace{0pt}
\end{minipage}\HR
```

Sometimes it is helpful to predefine the vertical dimension of a paragraph box. For this purpose today's LATEX offers additional optional arguments for `\parbox` and the minipage environment.

[1] TEX uses `\prevdepth` to calculate the interline space needed and −1000pt indicates that this space should be suppressed.

\parbox[*pos*][*height*][*inner-pos*]{*width*}{*text*}
\begin{minipage}[*pos*][*height*][*inner-pos*]{*width*} *text* \end{minipage}

The *inner-pos* argument determines the position of the *text* within the box. It can be `t`, `c`, `b`, or `s`. If not specified, the value of *pos* will be used. You can think of *height* and *inner-pos* as the vertical equivalent of the *width* and *pos* arguments of a `\makebox`. If you use the `s` position, the *text* will be vertically stretched to fill the given *height*. Thus, in this case you are responsible for providing vertically stretchable space if necessary, using, for example, the `\vspace` command.

As with the other box commands you can use `\height`, `\totalheight`, and so on to refer to the natural dimensions of the box when specifying the optional argument.

xx	Some text on top. And a few lines on the bottom of the box.	This time a few lines on the top of the box. But only one line down here.	xx

```
\usepackage{calc}
xx \fbox{\parbox[b][\height+\baselineskip][s]
        {20mm}{Some text on top. \par\vfill
               And a few lines on the
               bottom of the box.}}
   \fbox{\parbox[b][\height+\baselineskip][s]
        {20mm}{This time a few lines on the
               top of the box. But only one
               line \par\vfill down here.}} xx
```
A-2-14

A.2.3 Rule boxes

LaTeX's rule boxes are drawn with the `\rule` command:

\rule[*lift*]{*width*}{*height*}

If we write `\rule[4pt]{2cm}{1mm}` then we get a 2 cm long rule that is 1 mm thick and raised 4 pt above the baseline: ▬▬▬. The `\rule` command can also be used to construct rule boxes with zero width, that is invisible rules (also called *struts*). These struts are useful if you need to control the height or width of a given box (for example, to increase the height of a box framed with `\fbox` or `\framebox`, or to adjust locally the distance between rows in a table). Compare the following:

x111x $\boxed{\text{some text}}$ x222x $\boxed{\text{more text}}$ x333x

```
x111x
  \fbox{some text}
x222x
  \fbox{\rule[-5mm]{0cm}{15mm}more text}
x333x
```
A-2-15

As mentioned earlier, LaTeX makes boxes (including rules) behave like characters. For example, if used outside a paragraph they automatically start a new paragraph. With rules this is not always the desired behavior. To get a rule between two paragraphs, for instance, we have to use \noindent to suppress a paragraph indentation; otherwise, the line would be indented and stick out to the right.

... Some text for our page that might get reused over and over again.

A following paragraph. Some text for our page that might get reused over and over again.

A-2-16

```
\newcommand\sample{ Some text for our page
 that might get reused over and over again.}
\ldots \sample                        \par
\noindent\rule{\linewidth}{0.4pt} \par
A following paragraph. \sample
```

Due to this behavior the rule sits on the baseline of a one-line paragraph and is therefore visually much closer to the following paragraph. To place it at equal distance between the two lines, one could use the optional *lift* argument, but determining the right value (roughly 2.5 pt in this particular case) remains a matter of trial and error.

One solution is to suppress the generation of interline space, using the low-level TEX command \nointerlineskip, and to add the necessary spaces explicitly as shown in the next example. This time we omit \noindent so that the rule is indented by \parindent, and we use calc to calculate the rule width such that it leaves a space of size \parindent on the right as well.

... Some text for our page that might get reused over and over again.

A following paragraph. Some text for our page that might get reused over and over again.

A-2-17

```
\usepackage{calc}   % \sample as before
\ldots \sample                          \par
\nointerlineskip \vspace{5.8pt}
\rule{\linewidth-2\parindent}{0.4pt}\par
\nointerlineskip \vspace{5.8pt}
A following paragraph. \sample
```

The sum of the vertical spaces used plus the height of the rule amounts to 12 points (i.e., \baselineskip). However, this does not make the baselines of the two paragraphs 12 points apart; rather, it makes the distance from the bottom of the last line in the first paragraph (i.e., as produced by the "g" in "again") to the top of the first line in the next paragraph (i.e., as produced by the "A") be 12 points. Thus, if the text baselines should preferably fall onto a grid, a variant of Example A-2-16 using the optional *lift* argument is more appropriate.

Instead of using \rule together with \nointerlineskip, package or class writers often use the primitive TEX rule commands. They have the advantage of automatically suppressing interline space and do not require you to specify all dimensions. On the downside, they have an unusual syntax and cannot be used if the rule needs horizontal or vertical shifting, as in the previous example.

\hrule height *height* depth *depth* width *width* \relax
\vrule height *height* depth *depth* width *width* \relax

The \hrule primitive can only be used between paragraphs, while the \vrule primitive has to appear within paragraphs. If encountered in the wrong place,

	width	height	depth
\hrule	*	0.4pt	0.0pt
\vrule	0.4pt	*	*

Table A.4: Default values for TeX's rule primitives

the commands stop or start a paragraph as necessary. The commands can be followed by one or more of the keywords height, depth, and width together with a dimension value. Any order is allowed, and missing keywords get the defaults shown in Table A.4. An asterisk in that table means that the rule will extend to the boundary of the outer box. The \relax command at the end is not required but ensures that TeX knows that the rule specification has ended and will not misinterpret words in the text as keywords.

In the next example we use the default value for \hrule, resulting in a rule of 0.4pt height running through the whole galley width (since this is effectively the next outer box).

... Some text for our page that might get reused over and over again.

A following paragraph. Some text for our page that might get reused over and over again.

```
% \sample as before
\ldots \sample                    \par
\vspace{3pt}\hrule\relax\vspace{3pt}
A following paragraph. \sample    \par
```

A-2-18

A.2.4 Manipulating boxed material

Material can be typeset once and then stored inside a named box, whose contents can later be retrieved.

\newsavebox{*cmd*}	Declare box
\sbox{*cmd*}{*text*}	Fill box
\savebox{*cmd*}[*width*][*pos*]{*text*}	Fill box
\usebox{*cmd*}	Use contents

The command \newsavebox globally declares a command *cmd* (for example, \mybox), which can be thought of as a named bin. Typeset material can be stored there for later (multiple) retrieval.

The \sbox and \savebox commands are similar to \mbox and \makebox, except that they save the constructed box in the named bin (previously allocated with \newsavebox) instead of directly typesetting it. The \usebox command then allows the nondestructive use of the material stored inside such named bins. You can reuse the same bin (e.g., \mybox) several times within the scope of the current environment or brace group. It will always contain what was last stored in it.

Be careful not to use the command name \mybox directly, since it contains only the TeX number of the box in question. As a consequence, \mybox on its own will merely typeset the character at the position corresponding to the box number in the current font. Thus, you should manipulate boxes exclusively using the commands described above.

x1x inside box a x2x inside box b x3x
A-2-19 x1x inside box a x2x inside box b x3x

```
\newsavebox{\myboxa}\newsavebox{\myboxb}
\sbox{\myboxa}{inside box a}
\savebox{\myboxb}[2cm][l]{inside box b}
  x1x \usebox{\myboxa} x2x \usebox{\myboxb} x3x
\savebox{\myboxb}[2cm][r]{inside box b}
  \par
  x1x \usebox{\myboxa} x2x \usebox{\myboxb} x3x
```

In addition to the above commands, there exists the `lrbox` environment with the following syntax:

`\begin{lrbox}{`*cmd*`}` *text* `\end{lrbox}`

Here *cmd* should be a box register previously allocated with `\newsavebox`. The environment `lrbox` will save the *text* in this box for later use with `\usebox`. Leading and trailing spaces are ignored. Thus, `lrbox` is basically the environment form of `\sbox`. You can make good use of this environment if you want to save the body of some environment in a box for further processing. For example, the following code defines the environment `fcolumn`, which works like a column-wide `minipage` but surrounds its body with a frame.

A-2-20 | In this environment verbatim text like `\fcolbox` can be used. |

```
\usepackage{calc}
\newsavebox{\fcolbox} \newlength{\fcolwidth}
\newenvironment{fcolumn}[1][\linewidth]
  {\setlength{\fcolwidth}{#1-2\fboxsep-2\fboxrule}%
   \begin{lrbox}{\fcolbox}\begin{minipage}{\fcolwidth}}
  {\end{minipage}\end{lrbox}\noindent\fbox{\usebox{\fcolbox}}}
\begin{fcolumn}   In this environment verbatim text like
  \verb=\fcolbox= can be used.                  \end{fcolumn}
```

The above definition is interesting in several respects. The environment is defined with one optional argument denoting the width of the resulting box (default `\linewidth`). On the next line we calculate (using the calc package) the internal line length that we have to pass to the `minipage` environment. Here we have to subtract the extra space added by the `\fbox` command on both sides. Then the `lrbox` and `minipage` environments are started to typeset the body of the `fcolumn` environment into the box `\fcolbox`. When the end of the environment is reached those environments are closed. Then the `\fcolbox` is typeset inside an `\fbox` command. The `\noindent` in front suppresses any indentation in case the environment is used at the beginning of a paragraph or forms a paragraph by itself.

The `boxedminipage` described in Section 10.1.1 on page 595 can be implemented in a similar fashion. The only essential difference from the previous code is that we omit `\noindent` and pass the width as a mandatory argument and the position as an optional argument.

```
\usepackage{calc}
\newsavebox{\fcolbox} \newlength{\fcolwidth}
\newenvironment{boxedminipage}[2][c]
  {\setlength{\fcolwidth}{#2-2\fboxsep-2\fboxrule}%
   \begin{lrbox}{\fcolbox}%
   \begin{minipage}[#1]{\fcolwidth}}
  {\end{minipage}\end{lrbox}\fbox{\usebox{\fcolbox}}}
  left \begin{boxedminipage}[b]{4cm}
         In this environment verbatim text like
          \verb=\fcolbox= can be used.
       \end{boxedminipage}
  right
```

left | In this environment verbatim text like `\fcolbox` can be used. | right

A-2-21

If you compare this definition with the actual code in the package (which originates in LaTeX 2.09), it will be apparent that the coding features offered with the current version of LaTeX have their advantages.

A.2.5 Box commands and color

Even if you do not intend to use color in your own documents, by taking note of the points in this section you can ensure that your class or package is compatible with the color package. This may benefit people who choose to use your class or package together with the color package extensions.

The simplest way to ensure "color safety" is to always use LaTeX box commands rather than TeX primitives—that is, to use `\sbox` rather than `\setbox`, `\mbox` rather than `\hbox`, and `\parbox` or the `minipage` environment rather than `\vbox`. The LaTeX box commands have new options that make them as powerful as the TeX primitives.

As an example of what can go wrong, consider that in `{\ttfamily` *text*`}` the font is restored just *before* the `}`, whereas in the similar-looking construct `{\color{green}` *text*`}` the color is restored just *after* the final `}`. Normally, this distinction does not matter. But consider a primitive TeX box assignment such as

```
\setbox0=\hbox{\color{green} some text}
```

Now the color-restore operation occurs after the `}` and so is *not* stored in the box. Exactly which bad effects this introduces will depend on how color is implemented: the problems can range from getting the wrong colors in the rest of the document to causing errors in the `dvi` driver used to print the document.

Also of interest is the command `\normalcolor`. This is normally just `\relax` (i.e., does nothing), but you can use it like `\normalfont` to set regions of the page, such as captions or section headings, to the "main document color".

A.3 Control structure extensions

A.3.1 calc—Arithmetic calculations

The package calc (by Kresten Thorup and Frank Jensen) contains a set of macros for enhanced arithmetic in LaTeX. Usual arithmetic in TeX is done by simple low-level operations like \advance and \multiply. This package defines an infix notation arithmetic for LaTeX. In fact, it reimplements the LaTeX commands \setcounter, \addtocounter, \setlength, and \addtolength so that they can accept integer and length expressions rather than simple numbers and lengths.

An integer expression can contain integer numbers, TeX's integer registers, LaTeX's counters (e.g., \value{ctr}), parentheses, and binary operators (-, +, *, /). For instance, to advance a counter by five:

```
\usepackage{calc}    \newcounter{local}
\setcounter{local}{2} % initial setting for the example
The value is currently ``\thelocal''.\\
\setcounter{local}{\value{local}+5}
The value has now changed to ``\thelocal''.
```

A-3-1

The value is currently "2".
The value has now changed to "7".

An example is the definition of a command to print the time (note that the TeX register \time contains the number of minutes since midnight):

```
\usepackage{calc}
\newcounter{hours}\newcounter{minutes}
\newcommand\printtime{\setcounter{hours}{\time/60}%
        \setcounter{minutes}{\time-\value{hours}*60}%
        \thehours h \theminutes min}
The time is \printtime.
```

A-3-2

The time is 18h 53min.

When dealing with lengths, the subexpressions that are added or subtracted must be of the same type. That is, you cannot have "2cm+4", but an expression like "2cm+4pt" is legal because both subexpressions have dimensions. You can only divide or multiply by integers, so "2cm*4" is a legal subexpression but "2cm*4pt" is forbidden. Also, the length part must come first in an expression; thus, "4*2cm" is not allowed.

The commands described above allow you to calculate the width of one column in an *n*-column layout using the following single command (supposing that the variable *n* is stored as the first argument of a LaTeX macro):

```
\setlength\linewidth{(\textwidth-\columnsep*(#1-1))/#1}
```

The restriction that you can only multiply and divide by integers has been relaxed for calculations on lengths (dimensions). Those operations are allowed with real numbers.

\real{*decimal constant*} \ratio{*length expression*}{*length expression*}

A real number can be represented in two forms: the first command converts the *decimal constant* into a form that can be used in a calc formula. The second form denotes the real number obtained by dividing the value of the first expression by the value of the second expression.

As an example, assume you want to scale a figure so that it occupies the full width of the page (\textwidth). If the original dimensions of the figure are given by the length variables \Xsize and \Ysize, then the height of the figure after scaling will be:

```
\setlength\newYsize{\Ysize*\ratio{\textwidth}{\Xsize}}
```

The calc package is used in many examples in this book. If you do not want to apply it, you need to express the code given in the examples in the form of primitive (LA)TeX constructs. For example, the setting of \fcolwidth on page 869 has to be translated from

```
\setlength\fcolwidth{#1-2\fboxsep-2\fboxrule}%
```

to the following statements:

```
\setlength\fcolwidth{#1}%
\addtolength\fcolwidth{-2\fboxsep}%
\addtolength\fcolwidth{-2\fboxrule}
```

Besides the fact that the infix notation provided by the calc package is certainly more readable (and much easier to modify), it contains constructs for division and multiplication that cannot be expressed with standard LaTeX constructs. For example, to express the \topmargin calculation from page 198, the following code is necessary:

```
\setlength\topmargin{297mm}
\addtolength\topmargin{-\textheight}
\divide\topmargin by 3                 % TeX calculation
\addtolength\topmargin{-1in}
\addtolength\topmargin{-\headheight}
\addtolength\topmargin{-\headsep}
```

A.3.2 ifthen—Advanced control structures

Sometimes you may want to typeset different material depending on the value of a logical expression. This is possible with the standard package ifthen (written by Leslie Lamport, and reimplemented for the current LaTeX version by David Carlisle), which defines commands for building control structures with LaTeX.

\ifthenelse{*test*}{*then-code*}{*else-code*}

If the condition *test* is true, the commands in the *then-code* part are executed. Otherwise, the commands in the *else-code* part are executed.

A simple form of a condition is the comparison of two integers. For example, if you want to translate a counter value into English:

```
\usepackage{ifthen}
\newcommand\toEng[1]{\arabic{#1}\textsuperscript{%
 \ifthenelse{\value{#1}=1}{st}{%
  \ifthenelse{\value{#1}=2}{nd}{%
   \ifthenelse{\value{#1}=3}{rd}{%
    \ifthenelse{\value{#1}<20}{th}}%
                          {\typeout{Value too high}}}}}}
This is the \toEng{section} section in the \toEng{chapter}
appendix.
```

A-3-3 This is the 3rd section in the 1st appendix.

The following example defines a command to print the time in short form. It shows how complex operations (using the calc package) can be combined with conditional control statements.

```
\usepackage{ifthen,calc}
\newcounter{hours}\newcounter{minutes}
\newcommand{\Printtime}{\setcounter{hours}{\time/60}%
      \setcounter{minutes}{\time-\value{hours}*60}%
      \ifthenelse{\value{hours}<10}{0}{}\thehours:%
      \ifthenelse{\value{minutes}<10}{0}{}\theminutes}
The current time is ``\Printtime''.
```

A-3-4 The current time is “18:53”.

\equal{*string1*}{*string2*}

The \equal command evaluates to *true* if the two strings *string1* and *string2* are equal after they have been completely expanded. You should be careful when using fragile commands in one of the strings; they need protection with the \protect command.

```
\usepackage{ifthen,shortvrb}  \MakeShortVerb\|
\newcommand\BB{\CC}\newcommand\CC{\DD}
\newcommand\DD{AA} \newcommand\EE{EE}
|\BB|=|\EE|?  \ifthenelse{\equal{\BB}{\EE}}{True}{False}.\par
|\BB|=|\CC|?  \ifthenelse{\equal{\BB}{\CC}}{True}{False}.\par
|\DD|=|\BB|?  \ifthenelse{\equal{\DD}{\BB}}{True}{False}.
```

\BB=\EE? False.
\BB=\CC? True.
A-3-5 \DD=\BB? True.

One application for the preceding command could be in the definition of a command for printing an item and for entering it in the index. In the case where it is defined, the index entry will be typeset in boldface; otherwise, it will appear in a normal face. We use an optional argument for the least frequently occurring situation of the definition.

```
\usepackage{ifthen}
\newcommand{\IX}[2][R]{\texttt{#2}%
  \ifthenelse{\equal{#1}{D}}%
    {\index{#2|textbf}}{\index{#2}}}
  we define item \IX[D]{AAAA}
  \ldots{}     we reference item \IX{AAAA}
```

we define item AAAA ... we reference item AAAA

A-3-6

This gives the required visual representation in the .idx file by specifying entries of the following type:

```
\indexentry{AAAA|textbf}{874}    \indexentry{AAAA}{874}
```

A more complicated example, where you have complete control of what goes or does not go into the index or in the text, involves the extended index command `\IXE`, defined in the following example. Its default optional argument "`!*!,!`" contains a string that you will probably never want to use in the text (we hope). If you use the command `\IXE` with only one (normal) argument, then you will enter the same information into the index and the text. By specifying an optional argument, you can enter something in the index that is different from what is printed in the text. All possible combinations are shown below. The vertical bars around the commands show that no unwanted spaces are generated.

```
\usepackage{ifthen}
\newcommand\IXE[2][!*!,!]{%
 \ifthenelse{\equal{#1}{!*!,!}}%
   {\ifthenelse{\equal{#2}{}}{}{\textbf{#2}\index{#2}}}%
   {\ifthenelse{\equal{#1}{}}{}{\index{#1}}%
    \ifthenelse{\equal{#2}{}}{}{\textbf{#2}}}}
\par Identical in text and index |\IXE{both}|.
\par Different in text and index |\IXE[index]{text}|.
\par Only to index |\IXE[indexonly]{}|.
\par In text only |\IXE[]{textonly}|.
\par Nothing in text or index |\IXE[]{}|.
```

Identical in text and index |**both**|.
Different in text and index |**text**|.
Only to index ||.
In text only |**textonly**|.
Nothing in text or index ||.

A-3-7

The .idx file contains only three entries, since the case with the empty optional argument "`[]`" does not generate an index entry:

```
\indexentry{both}{874}
\indexentry{index}{874}
\indexentry{indexonly}{874}
```

	TeX switches (can only be queried)
`hmode`	`true`, if typesetting is done in a horizontal direction (e.g., inside a paragraph or an LR box).
`vmode`	`true`, if typesetting is done vertically (e.g., if TeX is between paragraphs).
`mmode`	`true`, if TeX is typesetting a formula.
	LaTeX switches (last two can be set)
`@twoside`	`true`, if LaTeX is typesetting for double-sided printing.
`@twocolumn`	`true`, if LaTeX is typesetting in standard two-column mode (`false` inside `multicols` environments).
`@firstcolumn`	`true`, if `@twocolumn` is `true` and LaTeX is typesetting the first column.
`@newlist`	`true`, if LaTeX is at the beginning of a list environment (will be set to `false` when text *after* the first `\item` command is encountered).
`@inlabel`	`true`, after an `\item` command until the text following it is encountered.
`@noskipsec`	`true`, after a run-in heading until the text following it is encountered.
`@afterindent`	Switch checked by command `\@afterheading` (usually used in headings) to prevent (if `false`) indentation of next paragraph.
`@tempswa`	Temporary switch used internally by many LaTeX commands to communicate with each other.

Table A.5: LaTeX's internal `\boolean` switches

`\boolean{`*string*`}` `\newboolean{`*string*`}` `\setboolean{`*string*`}{`*value*`}`

Basic TeX knows about some switches that can have the value `true` or `false`.[1] To define your own switch, use `\newboolean` where *string* is a sequence of letters. This switch is initially set to `false`. To change its value, use `\setboolean` where the *value* argument is either the string `true` or `false`. You can then test the value by using `\boolean` in the first argument of `\ifthenelse`. It is also possible to test all such internal flags of LaTeX with this command (the most common ones are shown in Table A.5). An example could be a test to see whether a document is using a one- or two-sided layout.

A-3-8

Two-sided printing.

```
\usepackage{ifthen}
\ifthenelse{\boolean{@twoside}}{Two-sided}{One-sided} printing.
```

`\lengthtest{`*test*`}`

To compare dimensions, use `\lengthtest`. In its *test* argument you can compare two dimensions (either explicit values like `20cm` or names defined by `\newlength`) using one of the operators <, =, or >.

[1] In the LaTeX kernel they are normally built using the more primitive `\newif` command.

As an example, let us consider a figure characterized by its dimensions \Xsize and \Ysize. It should be made to fit into a rectangular area with dimensions \Xarea and \Yarea, but without changing the aspect ratio of the figure. The following code calculates the new dimensions of the figure (\newX and \newY). The trick is to first calculate and compare the aspect ratios of both the rectangle and the figure, and then to use the result to obtain the magnification factor.

```
\newlength{\sizetmp}\newlength{\areatmp}
\setlength\sizetmp{1pt*\ratio{\Xsize}{\Ysize}}
\setlength\areatmp{1pt*\ratio{\Xarea}{\Yarea}}
\ifthenelse{\lengthtest{\sizetmp > \areatmp}}%
 {\setlength\newX{\Xarea}\setlength\newY{\newX*\ratio{\Ysize}{\Xsize}}}
 {\setlength\newY{\Yarea}\setlength\newX{\newY*\ratio{\Xsize}{\Ysize}}}
```

\isodd{*number*}

With the \isodd command you can test whether a given *number* is odd. If, for example, the string generated by a \pageref command is a valid number (as it normally is), then you can use the command in the following way:

This is an even-numbered page. 6	This is an odd-numbered page. 7

```
\usepackage{ifthen} \newcounter{pl}
\newcommand\pcheck{\stepcounter{pl}\label{pl-\thepl}%
 \ifthenelse{\isodd{\pageref{pl-\thepl}}}{odd}{even}}
This is an \pcheck-numbered page.  \newpage
This is an \pcheck-numbered page.
```

A-3-9

The \isodd command is specially tailored to support the above application even though the result of \pageref might be undefined in the first LaTeX run. Note that you cannot omit the \label and \pageref and instead simply use \thepage. The reason is that pages are built asynchronously. As a consequence, your code might get evaluated while a page is being built, and later on LaTeX's output routine might decide to move that bit of the text to the next page, making the evaluation invalid if \thepage were used.

\whiledo{*test*}{*do-clause*}

The \whiledo command is valuable for executing certain repetitive command sequences. The following simple example shows how the command works:

I should not talk during seminar (1). I should not talk during seminar (2). I should not talk during seminar (3). I should not talk during seminar (4).

```
\usepackage{ifthen}  \newcounter{howoften}
\setcounter{howoften}{1}
\whiledo{\value{howoften}<5}{I should not talk
     during seminar (\thehowoften).
     \stepcounter{howoften}}
```

A-3-10

`\and` `\or` `\not` `\(` `\)`

Multiple conditions can be combined into logical expressions via the logical operators (`\or`, `\and`, and `\not`), using the commands `\(` and `\)` as parentheses. A simple example is seen below.

You agree “OK” or don’t “not OK”.
A-3-11 D’accord “OK” ou pas “not OK”?

```
\usepackage{ifthen}
\newcommand{\QU}[2]{%
 \ifthenelse{\(\equal{#1}{ENG}\and\equal{#2}{yes}\)
    \or      \(\equal{#1}{FRE}\and\equal{#2}{oui}\)}%
             {``OK''}{``not OK''}}
You agree \QU{ENG}{yes} or don't \QU{ENG}{no}. \par
D'accord  \QU{FRE}{oui} ou pas   \QU{FRE}{non}?
```

A.4 Package and class file structure

In this section we discuss what commands are available for the authors of package or class files. Even if you do not intend to write your own package, this section will help you understand the structure and content of class and package files like book or varioref, and thus help you to make better use of them.

The general structure of class and package files is identical and consists of the following parts:

⟨*identification*⟩
⟨*initial code*⟩
⟨*declaration of options*⟩
⟨*execution of options*⟩
⟨*package loading*⟩
⟨*main code*⟩

All these parts are optional. We discuss the commands available in each of the individual parts below. Table A.6 on page 879 gives a short overview.

A.4.1 The identification part

This part of a class or package file is used to define the nature of the file and may also state the LaTeX 2_ε distribution release minimally required.

`\ProvidesClass{`*name*`}[`*release information*`]`

A class file identifies itself with a `\ProvidesClass` command. The argument *name* corresponds to the name of the class as it will be used in the mandatory argument of the `\documentclass` command (i.e., the file name without an extension). The optional argument *release information*, if present, should begin with a date in the form YYYY/MM/DD, separated with a space from the version number or identification, followed optionally by some text describing the class. For example, the class report contains something like

```
\ProvidesClass{report}[2001/04/21 v1.4e Standard LaTeX document class]
```

In a document you can make use of the *release information* by specifying the date as a second optional argument to the `\documentclass` command as follows:

```
\documentclass[twocolumn]{report}[2001/04/21]
```

This enables LaTeX to check that the report class used has at least a release date of 2001/04/21 or is newer. If the class file is older, a warning is issued. Thus, if you make use of a new release of a class file and send your document to another site, the people there will be informed if their LaTeX distribution is out of date.

`\ProvidesPackage{`*name*`}[`*release information*`]`

This command identifies a package file. The structure is the same as for the `\ProvidesClass` command. Again, the date in the *release information* can be used in a second optional argument to `\usepackage` to ensure that an up-to-date version of the package file is loaded. For example:

```
\usepackage[german]{varioref}[2001/09/01]
```

`\ProvidesFile{`*filename*`}[`*release information*`]`

This command identifies any other type of file. For this reason *filename* must contain the full file name including the extension.

`\NeedsTeXFormat{`*format*`}[`*release*`]`

In addition to one of the above commands, the ⟨*identification*⟩ part usually contains a `\NeedsTeXFormat` declaration. The *format* must be the string `LaTeX2e`. If the optional *release* argument is specified, it should contain the release date of the required LaTeX 2_ε distribution in the form YYYY/MM/DD. For example,

```
\NeedsTeXFormat{LaTeX2e}[2001/06/01]
```

would require at least the LaTeX 2_ε release distributed on June 1, 2001. If this command is present, anyone who tries to use your code together with an older LaTeX release will receive a warning message that something might fail. A newer release date is accepted without a warning.

All four declarations are optional. Nevertheless, their use in distributed class and package files will ease the maintenance of these files.

Identification part
`\NeedsTeXFormat{`*format*`}[`*release*`]` Needs to run under *format* (`LaTeX2e`) with a release date not older than *release*
`\ProvidesClass{`*name*`}[`*release info*`]` `\ProvidesPackage{`*name*`}[`*release info*`]` Identifies class or package *name* and specifies *release information*
`\ProvidesFile{`*name*`}[`*release info*`]` Identifies other file *name* (with extension) and specifies *release information*
Declaration of options
`\DeclareOption{`*option*`}{`*code*`}` Declares *code* to be executed for *option*
`\PassOptionsToPackage{`*option-list*`}{`*package-name*`}` Passes *option-list* to *package-name*
`\DeclareOption*{`*code*`}` Declares *code* to be executed for any unknown option
`\CurrentOption` Refers to current option for use in `\DeclareOption*`
Execution of options
`\ExecuteOptions{`*option-list*`}` Executes code for every option listed in *option-list*
`\ProcessOptions` `\ProcessOptions*` Processes specified options for current class or package; starred form obeys the specified order
Package loading
`\RequirePackage[`*option-list*`]{`*package*`}[`*release*`]` Loads *package* with given *option-list* and a release date not older than *release*
Special commands for package and class files
`\AtEndOfPackage{`*code*`}` `\AtEndOfClass{`*code*`}` Defers execution of *code* to end of current package or class
`\AtBeginDocument{`*code*`}` `\AtEndDocument{`*code*`}` Executes *code* at `\begin{document}` or `\end{document}`
`\IfFileExists{`*file*`}{`*then-code*`}{`*else-code*`}` Executes *then-code* if *file* exists, *else-code* otherwise
`\InputIfFileExists{`*file*`}{`*then-code*`}{`*else-code*`}` If *file* exists, executes *then-code* and then inputs *file*; otherwise executes *else-code*
Special class file commands
`\LoadClass[`*option-list*`]{`*class*`}[`*release*`]` Like `\RequirePackage` for class files, but does not see global options if not explicitly passed to it
`\PassOptionsToClass{`*option-list*`}{`*class*`}` Passes *option-list* to *class*
`\OptionNotUsed` For use in `\DeclareOption*` if necessary

Table A.6: Commands for package and class files

A.4.2 The initial code part

You can specify any valid LaTeX code in the ⟨*initial code*⟩ part, including code that loads packages with the `\RequirePackage` command (see Section A.4.5) if their code is required in one of the option declarations. For example, you might want to load the calc package at this point, if you plan to use it later. However, normally this part is empty.

A.4.3 The declaration of options

In this part all options known to the package or class are declared using the `\DeclareOption` command. It is forbidden to load packages in this part.

`\DeclareOption{`*option*`}{`*code*`}`

The argument *option* is the name of the option being declared and *code* is the code that will execute if this option is requested. For example, the paper size option `a4paper` normally has a definition of the following form:

```
\DeclareOption{a4paper}{\setlength\paperheight{297mm}%
                        \setlength\paperwidth{210mm}}
```

In principle, any action—from setting a flag to complex programming instructions—is possible in the *code* argument of `\DeclareOption`.

An important function for use in `\DeclareOption` is the command `\PassOptionsToPackage`. It can pass one or more options to some other package that is loaded later.

`\PassOptionsToPackage{`*option-list*`}{`*package-name*`}`

The argument *option-list* is a comma-separated list of options that should be passed to the package with name *package-name* when it is loaded in the ⟨*package loading*⟩ part.[1] Suppose, for example, that you want to define a class file that makes use of two packages, say, A and B, both supporting the option `infoshow`. To support such an option in the class file as well, you could declare

```
\DeclareOption{infoshow}{%
    \PassOptionsToPackage{infoshow}{A}%
    \PassOptionsToPackage{infoshow}{B}%
    ⟨code to support infoshow in the class⟩}
```

If a package or class file is loaded with an option that it does not recognize, it will issue a warning (in case of a package file) or silently ignore the option (in case of a class file), assuming that it is a global option to be passed to other packages

[1] It is the responsibility of the package writer to actually load such packages. LaTeX does not check that packages receiving options via `\PassOptionsToPackage` are actually loaded later on.

subsequently loaded with \usepackage. However, this behavior is not hard-wired and can be modified using a \DeclareOption* declaration.

\DeclareOption*{*code*}

The argument *code* specifies the action to take if an unknown option is specified on the \usepackage or \RequirePackage command. Within this argument \CurrentOption refers to the name of the option in question. For example, to write a package that extends the functionality of some other package, you could use the following declaration:

Command does not act on global options!

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{A}}
```

This would pass all options not declared by your package to package A. If no \DeclareOption* declaration is given, the default action, described above, will be used.

By combining \DeclareOption* with \InputIfFileExists (see below), you can even implement conditional option handling. For example, the following code tries to find files whose names are built up from the option name:

```
\DeclareOption*{\InputIfFileExists{g-\CurrentOption.xyz}{}%
    {\PackageWarning{somename}{Option \CurrentOption\space
                                              not recognized}}}
```

If the file g-*option*.xyz can be found, it will be loaded; otherwise, the option is ignored with a warning.

A.4.4 The execution of options

Two types of actions are normally carried out after all options are declared. You might want to set some defaults, such as the default paper size. Then the list of options specified needs to be examined and the code for each such option needs to be executed.

\ExecuteOptions{*option-list*}

The \ExecuteOptions command executes the code for every option listed in *option-list* in the order specified. It is just a convenient shorthand to set up defaults by executing code specified earlier with a \DeclareOption command. For example, the standard class book issues something similar to

```
\ExecuteOptions{letterpaper,twoside,10pt}
```

to set up the defaults. You can also use \ExecuteOptions when declaring other options, such as a definition of an option that automatically implies others. The \ExecuteOptions command can be used only prior to executing the

\ProcessOptions command because, as one of its last actions, the latter command reclaims all of the memory taken up by the code for the declared options.

```
\ProcessOptions
```

When the \ProcessOptions command is encountered, it examines the list of options specified for this class or package and executes the corresponding code. More precisely, when dealing with a package the global options (as specified on the \documentclass command) and the directly specified options (the optional argument to the \usepackage or \RequirePackage command) are tested. For every option declared by the package, the corresponding code is executed. This execution occurs in the same order in which the options were specified by the \DeclareOption declarations in the package, not in the order in which they appear on the \usepackage command. Global options that are not recognized are ignored. For all other unrecognized options the code specified by \DeclareOption* is executed or, if this declaration is missing, an error is issued.

Thus, packages that use only \DeclareOption* when declaring options will not act upon global options specified on the \documentclass, but rather will accept only those that are explicitly given on the \usepackage or \RequirePackage declaration.

In the case of a class file, the action of \ProcessOptions is the same without the added complexity of the global options.

Preventing unwanted expansion

There is one potential problem when using \ProcessOptions: the command searches for a following star (even on subsequent lines) and thereby may incorrectly expand upcoming commands following it. To avoid this danger use \relax at the end to stop the search immediately and start the execution of the options.

```
\ProcessOptions*
```

For some packages it may be more appropriate if they process their options in the order specified on the \usepackage command rather than using the order given through the sequence of \DeclareOption commands. For example, in the babel package, the last language option specified is supposed to determine the main document language. Such a package can execute the options in the order specified by using \ProcessOptions* instead of \ProcessOptions.

A.4.5 The package loading part

Once the options are dealt with, it might be time to load one or more additional packages—for example, those to which you have passed options using \PassOptionsToPackage.

\RequirePackage [*option-list*] {*package*} [*release*]

This command is the package/class counterpart to the document command \usepackage. If *package* was not loaded before, it will be loaded now with the options specified in *option-list*, the global options from the `\documentclass` command, and all options passed to this package via `\PassOptionsToPackage`.

LaTeX loads a package only once because in many cases it is dangerous to execute the code of a package several times. Thus, if you require a package with a certain set of options, but this package was previously loaded with a different set not including all options requested at this time, then the user of your package has a problem. In this situation LaTeX issues an error message informing users of your package about the conflict and suggesting that they load the package with a `\usepackage` command and all necessary options.

The optional *release* argument can be used to request a package version not older than a certain date. For this scheme to work, the required package must contain a `\ProvidesPackage` declaration specifying a release date.

`\RequirePackageWithOptions{`*package*`}[`*release*`]`

This command works like `\RequirePackage` except that the options passed to it are exactly those specified for the calling package or class. This facilitates the generation of variant packages that take exactly the same set of options as the original. See also the discussion of `\LoadClassWithOptions` on page 887.

A.4.6 The main code part

This final part of the file defines the characteristics and implements the functions provided by the given class or package. It can contain any valid LaTeX construct and usually defines new commands and structures. It is good style to use standard LaTeX commands, as described in this appendix, such as `\newlength`, `\newcommand`, `\CheckCommand`, and so on, rather than relying on primitive TeX commands, as the latter do not test for possible conflicts with other packages.

A.4.7 Special commands for package and class files

`\AtEndOfPackage{`*code*`}`   `\AtEndOfClass{`*code*`}`

Sometimes it is necessary to defer the execution of some code to the end of the current package or class file. The above declarations save the *code* argument and execute it when the end of the package or class is reached. If more than one such declaration is present in a file, the *code* is accumulated and finally executed in the order in which the declarations were given.

`\AtBeginDocument{`*code*`}`   `\AtEndDocument{`*code*`}`

Other important points at which you might want to execute deferred code are the beginning and the end of the document or, more exactly, the points where the `\begin{document}` and `\end{document}` are processed. The above commands allow packages to add code to this environment without creating any conflicts with other packages trying to do the same.

Note, however, that code in the `\AtBeginDocument` hook is part of the preamble. Thus, restrictions limit what can be put there; in particular, no typesetting can be done.

`\IfFileExists{`*file*`}{`*then-code*`}{`*else-code*`}`
`\InputIfFileExists{`*file*`}{`*then-code*`}{`*else-code*`}`

If your package or class tries to `\input` a file that does not exist, the user ends up in TeX's file-error loop. It can be exited only by supplying a valid file name. Your package or class can avoid this problem by using `\IfFileExists`. The argument *file* is the file whose existence you want to check. If this *file* is found by LaTeX, the commands in *then-code* are executed; otherwise, those in *else-code* are executed. The command `\InputIfFileExists` not only tests whether *file* exists, but also inputs it immediately after executing *then-code*. The name *file* is then added to the list of files to be displayed by `\listfiles`.

`\PackageWarning{`*name*`}{`*warning-text*`}`
`\PackageWarningNoLine{`*name*`}{`*warning-text*`}`
`\PackageInfo{`*name*`}{`*info-text*`}`

When a package detects a problem it can alert the user by printing a warning message on the terminal. For example, when the multicol package detects that `multicols*` (which normally generates unbalanced columns) is used inside a box, it issues the following warning:[1]

```
\PackageWarning{multicol}{multicols* inside a box does
 not make sense.\MessageBreak  Going to balance anyway}
```

This will produce a warning message, which is explicitly broken into two lines via the `\MessageBreak` command:

```
Package multicol Warning: multicols* inside a box does not make sense.
(multicol)                Going to balance anyway on input line 6.
```

The current line number is automatically appended. Sometimes it would be nice to display the current file name as well, but unfortunately this information is not available on the macro level.

Depending on the nature of the problem, it might be important to tell the user the source line on which the problem was encountered. In other cases this information is irrelevant, such as when the problem happens while the package is being loaded. In this situation `\PackageWarningNoLine` should be used; it produces the same result as `\PackageWarning` but omits the phrase "on input line *num*".

[1] In a box, balancing is essential since a box can grow arbitrarily in vertical direction, so all material would otherwise end up in the first column.

If the information is of lower importance and should appear just in the transcript file, then one can use \PackageInfo. For example, after loading the shortvrb package and issuing the declaration \MakeShortVerb\=, the transcript file will show the following:

```
Package shortvrb Info: Made = a short reference for \verb on input line 3.
```

A \PackageInfoNoLine command is not provided. If you really want to suppress the line number in an informational message, use \@gobble as the last token in the second argument of \PackageInfo.

\PackageError{*name*}{*short-text*}{*long-text*}

If the problem detected is severe enough to require user intervention, one can signal an error instead of a warning. If the error is encountered, the *short-text* is displayed immediately and processing stops. For example, if inputenc encounters an 8-bit character it does not recognize, it will produce the following error:

```
! Package inputenc Error: Keyboard character used is undefined
(inputenc)                in inputencoding 'latin1'.

See the inputenc package documentation for explanation.
Type  H <return>  for immediate help.
 ...

l.5 abc^^G
?
```

If the user then presses “h” or “H”, the *long-text* is offered. In this case it is:

```
You need to provide a definition with \DeclareInputText
or \DeclareInputMath before using this key.
```

As before, you can explicitly determine the line breaks in the error and help texts by using \MessageBreak.

\ClassWarning{*name*}{*warning-text*}
\ClassWarningNoLine{*name*}{*warning-text*}
\ClassInfo{*name*}{*info-text*}
\ClassError{*name*}{*short-text*}{*long-text*}

Information, warning, and error commands are not only available for packages—similar commands are provided for document classes. They differ only in the produced texts: the latter commands print “Class” instead of “Package” in the appropriate places.

```
% -------------------------------- identification -----------------------
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{myart}[1994/01/01]
% --------------------------------  initial code ------------------------
\RequirePackage{ifthen}     \newboolean{cropmarks}
% ---------------------------  declaration of options --
\DeclareOption{cropmarks}{\setboolean{cropmarks}{true}}
\DeclareOption{bind}     {\AtEndOfClass{\addtolength\oddsidemargin{.5in}%
                          \addtolength\evensidemargin{-.5in}}}
\DeclareOption*          {\PassOptionsToClass{\CurrentOption}{article}}
% ----------------------------  execution of options --------------------
\ProcessOptions \relax                              % cf. hint on p. 882!
% --------------------------------package loading -----------------------
\LoadClass{article}                                 % the real code
% -------------------------------- main code ----------------------------
\newenvironment{Notes}{...}{...}                    % the new environment
\ifthenelse{\boolean{cropmarks}}                    % support for cropmarks
   {\renewcommand{\ps@plain}{...} ...}{}
```

Figure A.1: An example of a class file extending article

A.4.8 Special commands for class files

It is sometimes helpful to build a class file as a customization of a given general class. To support this concept two commands are provided.

`\LoadClass`[*option-list*]{*class*}[*release*]

The `\LoadClass` command works like the `\RequirePackage` command with the following three exceptions:

- The command can be used only in class files.
- There can be at most one `\LoadClass` command per class.
- The global options are not seen by the *class* unless explicitly passed to it via `\PassOptionsToClass` or specified in the *option-list*.

`\PassOptionsToClass`{*option-list*}{*class*}

The command `\PassOptionsToClass` can be used to pass options to such a general class. An example of such a class file augmentation is shown in Figure A.1. It defines a class file `myart` that accepts two extra options, `cropmarks` (making crop marks for trimming the pages) and `bind` (shifting the printed pages slightly to the outside to get a larger binding margin), as well as one additional environment, `Notes`.

The cropmarks option is implemented by setting a Boolean switch and redefining various \pagestyles if this switch is true. The bind option modifies the values of \oddsidemargin and \evensidemargin. These length registers do not have their final values at the time the bind option is encountered (they are set later, when the article class is loaded by \LoadClass), so the modification is deferred until the end of the myart class file using the \AtEndOfClass command.

\OptionNotUsed

If your *code* for \DeclareOption* inside a class file is more complex (e.g., trying to handle some options but rejecting others), you might need to explicitly inform LaTeX that the option was not accepted with the help of the \OptionNotUsed command. Otherwise, LaTeX will think that the option was used and will not produce a warning if the option is not picked up by a later package.

\LoadClassWithOptions{*class*}[*release*]

This command is similar to \LoadClass, but it always calls the *class* with exactly the same option list that is being used by the current class, rather than the options explicitly supplied or passed on by \PassOptionsToClass. It is mainly intended to allow one class to build on another. For example:

```
\LoadClassWithOptions{article}
```

This should be contrasted with the following slightly different construction:

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions                    \LoadClass{article}
```

As used here, the effects are more or less the same, but the version using \LoadClassWithOptions is slightly quicker (and less onerous to type). If, however, the class declares options of its own, then the two constructions are different. Compare, for example,

```
\DeclareOption{landscape}{...}
\ProcessOptions                    \LoadClassWithOptions{article}
```

with:

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions                    \LoadClass{article}
```

In the first example, the article class will be called with the option landscape only when the current class is called with this option. In the second example, however, the option `landscape` will never be passed to the article class, because the default option handler only passes options that are *not* explicitly declared.

`\@ifpackageloaded{`*package*`}{`*true-code*`}{`*false-code*`}`
`\@ifpackagelater{`*package*`}{`*date*`}{`*true-code*`}{`*false-code*`}`
`\@ifpackagewith{`*package*`}{`*options*`}{`*true-code*`}{`*false-code*`}`

Sometimes it is useful to be able to find out if a package was already loaded, and if so, how. For this purpose, three commands are made available to class (and package) writers. To find out if a *package* has already been loaded, use `\@ifpackageloaded`. If it was loaded, the *true-code* is executed; otherwise, the *false-code* is executed. To find out if a *package* has been loaded with a version more recent than *date*, use `\@ifpackagelater`. Finally, to find out if a *package* has been loaded with at least the options in the (comma-separated) list *options*, use `\@ifpackagewith`.

The fontenc package cannot be tested with the above commands. That's because it pretends that it was never loaded to allow for repeated reloading with different options (see the file `ltoutenc.dtx` in the LaTeX distribution for details).

A.4.9 A minimal class file

Every class file *must* contain four things: a definition of `\normalsize`, values for `\textwidth` and `\textheight`, and a specification for page numbering. Thus, a minimal document class file[1] looks like this:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard LaTeX minimal class]
\renewcommand\normalsize{\fontsize{10pt}{12pt}\selectfont}
\setlength\textwidth{6.5in}
\setlength\textheight{8in}
\pagenumbering{arabic}       % needed even though this class will
                             % not show page numbers
```

This class file will, however, not support footnotes, marginals, floats, or other features. Naturally, most classes will contain more than this minimum!

[1]This class is in the standard distribution, as `minimal.cls`.

Appendix B

Tracing and Resolving Problems

In an ideal world all documents you produced would compile without problems and give high-quality output as intended. If you are that lucky, there will be no need for you to consult this appendix, ever. However, if you run into a problem of some kind, the material in this appendix should help you to resolve your problem easily.

We start with an alphabetical list of all error messages, those after which LaTeX stops and asks for advice. "All" in this context means all LaTeX kernel errors (their text starts with `LaTeX Error:`), practically all TeX errors (i.e., those directly produced by the underlying engine), and errors from the packages amsmath, babel, docstrip, calc, color, graphics, graphicx, inputenc, fontenc, and textcomp. Errors reported by other packages—those that identify themselves as

`! Package` ⟨*package*⟩ `Error:` ⟨*error text*⟩

where ⟨*package*⟩ is not one of the above—are not included. For such errors you should refer to the package description elsewhere in the book or consult the original package documentation.

But even if there are no real errors that stop the processing, warning and information messages might be shown on the terminal or in the transcript file. They are treated in Section B.2, where you will find all LaTeX core messages and all relevant TeX messages that may need your attention, together with an explanation of their possible causes and suggestions on how to deal with them.

The final section deals with tools for tracing problems in case the error or warning information itself is not sufficient or does not exist. We will explore ways to display command definitions and register values, then take a look at diagnosing and solving page-breaking problems. This is followed by suggestions for identifying and solving paragraph-breaking problems. We finish with a description of the trace package, which helps in thoroughly tracing command execution, in case your own definitions or those of others produce unexpected results.

Some of the material in this appendix can be considered "low-level" TeX, something that, to the authors' knowledge, has never been described in a "LaTeX" book. It is, however, often important information. Directing the reader to books like *The TeXbook* does not really help, since most of the advice given in books about plain TeX is not applicable to LaTeX or produces subtle errors when used. We therefore try to be as self-contained as possible by offering all relevant information about the underlying TeX engine as far as it makes sense within the LaTeX context.

B.1 Error messages

When LaTeX stops to display an error message, it also shows a line number indicating how far it got in the document source. However, because of memory considerations in the design of TeX itself, it does not directly show to which file this source line number belongs. For simple documents this is not a problem, but if your document is split over many files you may have to carefully look at the terminal output or the transcript file to identify the file LaTeX is currently working on when the error occurs.

Finding the source line of an error

Whenever LaTeX starts reading a file, it displays a "(" character that is immediately followed by the file name. Once LaTeX has finished reading the file, it displays the matching ")" character. In addition, whenever it starts preparing to output a page, it displays a "[" character followed by the current page number. Thus, if you see something like

```
(./trial.tex [1] (./ch-1.tex [2] [3] (./table-1.tex [4] [5]) [6]
! Undefined control sequence.
<argument> A \textss
                    {Test}
l.235 \section{A \textss{Test}}
                              \label{sec:test}
?
```

you can deduce that the error happened inside an argument of some command (`<argument>`) and was detected when LaTeX gathered material for page 7. It got as far as reading most of line 235 in the file `ch-1.tex`. In this example the error is readily visible in the source line: `\textsf` was misspelled as `\textss` inside the argument to the `\section` command. In some cases, however, the relationship between error and source line is blurred or even nonexistent.

For example, if you define `\renewcommand\thepart{\Alp{part}}`, then the typo will appear only when you use the `\part` command that executes your definition. In that case you get

```
! Undefined control sequence.
\thepart ->\Alp
               {part}
l.167 \part{Test}
```

In this particular case the actual error is not on line 167 and most likely not even in the current file—the `\part` command merely happens to call the faulty definition of `\thepart`.

Sometimes an error is detected by LaTeX while it is preparing a new page. Since this is an asynchronous operation, the source line listed in the error message is of no value whatsoever. So if you do not understand how the error should be related to the source line, you may well be right—there is, indeed, no relationship. Here is an example:

```
! Undefined control sequence.
\thepage ->\romen
                 {page}
l.33 T
      his is a sample text to fill the page.
```

One way to obtain additional information about an error (or information about how LaTeX intends to deal with it) is to reply ⟨*h*⟩ in response to the ? that follows the error message. If used with a TeX error such as the one above, we get

```
? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
```

You probably already see the problem with advice coming directly from the TeX engine: you may have to translate it, because it often talks about commands that are not necessarily adequate for LaTeX documents (e.g., for `\def` you should read `\newcommand` or `\renewcommand`). With real LaTeX errors this is not the case, though here you sometimes get advice that is also not really helpful:

```
You're in trouble here.  Try typing  <return>  to proceed.
If that doesn't work, type  X <return>  to quit.
```

Well, thank you very much, we already knew that! It is, however, worth a try, since there are many messages with more detailed advice.

Another way to get additional information about an encountered error is to set the counter `errorcontextlines` to a large positive value. In that case LaTeX will list the stack of the current macro executions:

```
1  ! Undefined control sequence.
2  \thepage ->\romen
3                    {page}
4  \@oddfoot ->\reset@font \hfil \thepage
5                                         \hfil
6  \@outputpage ...lor \hb@xt@ \textwidth {\@thefoot
7                                                  }\color@endbox }}\globa...
8
9  \@opcol ...lumn \@outputdblcol \else \@outputpage
10                                                 \fi \global \@mparbotto...
11 <output> ...specialoutput \else \@makecol \@opcol
12                                                 \@startcolumn \@whilesw...
13 <to be read again>
14                   T
15 l.33 T
16        his is a sample text to fill the page.
```

You read this bottom up: LaTeX has seen the `T` (lines 15 and 16) but wants to read it again later (`<to be read again>`, lines 13 and 14) because it switched to the output routine (`<output>`). There it got as far as executing the command `\@opcol` (lines 11 and 12), which in turn got as far as calling `\@outputpage` (lines 9 and 10), which was executing `\@thefoot` (lines 6 and 7). Line 4 is a bit curious since it refers to `\@oddfoot` rather than `\@thefoot` as one would expect (`\@thefoot` expands to `\@oddfoot`, so it is immediately fully expanded and not put onto the stack of partially expanded macros). Inside `\@oddfoot` we got as far as calling `\thepage`, which in turn expanded to `\romen` (lines 2 and 3), which is finally flagged as an undefined command (line 1).

Fortunately, in most cases it is sufficient only to display the error message and the source line. This is why LaTeX's default value for `errorcontextlines` is `-1`, which means not showing any intermediate context.

Errors can also occur when LaTeX is processing an intermediate file used to transfer information between two runs (e.g., `.aux` or `.toc` files). Data in such files can be corrupted due to an error that happened in a previous run. Even if you have corrected that error in your source, traces of it may still be present in such external files. Therefore, in some cases you may have to delete those files before running LaTeX again, although often the problem vanishes after another run.

Common sources for such nasty errors in LaTeX are so-called *fragile* commands used unprotected in *moving arguments*. Technically, a moving argument is an argument that is internally expanded by LaTeX without typesetting it directly (e.g., by using the internal LaTeX construct `\protected@edef`[1]). But as a rule of thumb you

[1] Some people have heard that the TeX primitive `\edef` exists for this purpose. It is not advisable to use it in your own commands, however, unless you know that it will never receive arbitrary document input. You should use `\protected@edef` instead, since that command prevents fragile commands from breaking apart if they are prefixed by `\protect`!

can think of it as an argument that is moved somewhere else before typesetting—for example, the arguments of sectioning commands, such as `\section` (sent to the table of contents), the argument of `\caption` (sent to the list of figures or tables), and the arguments of `\markboth` and `\markright`.

The best, though not very helpful, definition of a fragile command is that it is a command that produces errors if it is not preceded with a `\protect` command when used in a moving argument. Today, most common LaTeX commands have been made robust, so that such protection is not necessary. However, if you get strange errors from a command used in a moving argument, try preceding it with `\protect`. Typically, core LaTeX commands with optional arguments are fragile, but `\sqrt[3]{-1}` is robust and so are *all* user-defined commands with an optional argument. On the other hand, `\cong` is fragile in standard LaTeX, yet it becomes robust once the amsmath package is loaded. In other words, there are no precise rules defining which commands belong to which category. User-defined commands with only mandatory arguments are fragile if they contain any fragile commands in their definition. For example, the definition

```
\newcommand\frail{\ifthenelse{\value{section}<10 \and
                               \value{subsection}=1}%
                              {\typeout{Yes}}{\typeout{No}}}
```

is fragile because the comparison argument of `\ifthenelse` is fragile. If you used `\frail` in the `@` expression of a `tabular` (not that this makes much sense),

```
\nonstopmode      \begin{tabular}{@{\frail}l} x \end{tabular}
```

you would see the following 134 errors before LaTeX finally gives up (the left column displays the number of occurrences):

```
 1 ! Argument of \@array has an extra }.
 2 ! Argument of \@firstoftwo has an extra }.
 1 ! Extra }, or forgotten $.
 4 ! Extra }, or forgotten \endgroup.
 1 ! LaTeX Error:  Illegal character in array arg.
 1 ! LaTeX Error: Can be used only in preamble.
51 ! Misplaced \cr.
 2 ! Missing # inserted in alignment preamble.
 1 ! Missing = inserted for \ifnum.
49 ! Missing \cr inserted.
 2 ! Missing control sequence inserted.
 2 ! Missing number, treated as zero.
 1 ! Missing { inserted.
 2 ! Missing } inserted.
 1 ! Paragraph ended before \renew@command was complete.
 2 ! Paragraph ended before \reserved@b was complete.
 1 ! Paragraph ended before \reserved@c was complete.
```

```
2 ! Undefined control sequence.
1 ! Use of \@argtabularcr doesn't match its definition.
7 ! Use of \@array doesn't match its definition.
```

In fact, in this particular example TeX gets into a loop in which it tries to insert a `\cr` command, immediately rejects its own idea, and then repeats this process.

All TeX errors can be caused by a fragile command in a moving argument!

What we can learn from this example is the following: whenever you encounter a strange TeX error that has no simple explanation (e.g., a misspelled command name), it is possibly due to a fragile command that got broken in a moving argument—so try protecting it with `\protect` at the point where the error occurs. Since this can be the reason behind every TeX error, we shall not repeat this possible cause for every one of them (after all, more than 60 TeX error messages are explained below).

Errors produced by cross-reference keys

As discussed in Section A.1.1, a few restrictions are placed on the characters that can be used in reference key arguments of `\label` and `\bibitem`. In a nutshell, such keys sometimes act like moving arguments and, depending on the combination characters used and the packages loaded, all kinds of dreadful TeX errors may show up. In that case protection with using the `\protect` command will *not* work; instead, you have to use a simpler key conforming to the syntax restrictions for such keys.

Alphabetical listing of TeX and LaTeX errors

In the list of errors below, all TeX and all package errors are flagged with a boxed reference at the end of the error message. Unflagged error messages are LaTeX errors with the prefix "`LaTeX Error:`" omitted.

- * If LaTeX stops by just displaying a star, then it has reached the end of your source document without seeing a request to finish the job (i.e., `\end{document}` or `\stop`) and is now waiting for input from the terminal. While this is in itself not an error, in most circumstances it means that something went seriously wrong. If there have been no previous errors and your document finishes with `\end{document}`, then you might have forgotten to close a `verbatim` environment so that the remainder of the document was processed "verbatim".

  To find the source of this problem in a large document, reply `\end{foo}`, which either should give you an "Environment ... ended by..." error (indicating what environment LaTeX thinks is still open) or will be swallowed without any reaction, in which case you know that you are indeed in some "verbatim" context. In the latter event, try to interrupt LaTeX (by pressing Control-C or whatever your installation requires) and reply with "`x`" to the "Interruption" error to quit the job. Looking afterwards at the last page in the typeset document usually gives some hint about where things started to go wrong.

'⟨*character*⟩' invalid at this point calc

You loaded the calc package and one of the formulas in \setcounter, \setlength, \addtocounter, or \addtolength used a syntax not supported by calc. See Section A.3.1 for details.

⟨*command*⟩ allowed only in math mode amsmath

This command or environment can be used only in math mode. Check carefully to see what is missing from your document.

⟨*name*⟩ undefined

This error is triggered when you use \renewcommand for a ⟨*name*⟩ that is unknown to LaTeX. Either ⟨*name*⟩ was misspelled or you should have used \newcommand instead.

\< in mid line

The \<, defined within a tabbing environment, was encountered in the middle of a line. It can be used only at the beginning of a line (e.g., after \\).

A <Box> was supposed to be here TeX

This error is the result of using a box command, such as \sbox, with an invalid first argument (i.e., one not declared with \newsavebox). Usually, you first get the error "Missing number, treated as zero" indicating that TeX uses box register zero.

Accent ⟨*command*⟩ not provided by font family ⟨*name*⟩ textcomp

The textcomp package implements the TS1 encoding, which is unfortunately implemented fully by just a minority of the font families usable with LaTeX. No accent will be printed. See Section 7.5.4 for information on how to provide an alternative representation for it.

Argument of ⟨*command*⟩ has an extra } TeX

A right brace was used in place of a mandatory command argument (e.g., \mbox}). Fragile commands, when used without \protect in a moving argument, often break in a way that generates this or one of the other "extra" errors discussed below.

Bad \line or \vector argument

LaTeX issues this error if you specified a negative length or used an illegal slope with either \line or \vector. In the latter case, see Chapter 10 for alternatives.

Bad math environment delimiter

This error is triggered when a \(or \[command is encountered inside a formula, or when \) or \] is found in normal text. Check whether these commands are properly matched in your document.

\begin{⟨*env*⟩} allowed only in paragraph mode amsmath

There are many places, such as within LR-mode text or math mode, where it does not make sense to have a math display. With amsmath the whole display ⟨*env*⟩ will simply be ignored.

`\begin{⟨env⟩} on input line ⟨line number⟩ ended by \end{⟨other env⟩}`
You receive this error when LaTeX detects that the environment ⟨*env*⟩ was incorrectly terminated with the end-tag for the environment ⟨*other env*⟩. The most likely case is that you, indeed, forgot to close the environment ⟨*env*⟩.

Another possible source of this error is trying to use verbatim-like environments or an amsmath display environment inside the definition of your own environments, which is often impossible. See Section 3.4.3 on page 164 for solutions involving verbatim-like environments.

If neither is the case and you are absolutely sure that all environments are properly nested, then somewhere between the start of ⟨*env*⟩ and the point where the error was found there must be a command that issues an `\endgroup` without a prior matching `\begingroup` so that LaTeX is fooled into believing that the ⟨*env*⟩ environment ended at this point. One way to find that problem is to move the end-tag closer to the begin-tag, until the problem disappears.

`\begin{split} won't work here` [amsmath]
Either this `split` environment is not within an equation or perhaps you need to use `aligned` here.

`Can be used only in preamble`
LaTeX has encountered a command or environment that should be used only inside a package or the preamble (i.e., before `\begin{document}`). This error can also be caused by a second `\begin{document}`.

`Cannot be used in preamble`
Some commands—for example, `\nocite`—are allowed only in the document body (i.e., after `\begin{document}`). Move the declaration to that point.

`Cannot define Unicode char value < 00A0` [inputenc]
Values less than `"00A0` (decimal 160) are either invalid as Unicode values for text characters or must not be redefined in LaTeX.

`Cannot determine size of graphic in ⟨file⟩` [graphics/graphicx]
You did not specify an explicit image size on the `\includegraphics` command and LaTeX was unable to determine the image size from the graphics ⟨*file*⟩ directly. It usually does this automatically, for example, for `.eps` files by reading the bounding box information. However, depending on the graphics driver, it may be unable to extract this information from binary bitmap images such as `.jpg`, `.gif`, and `.png` files.

`Cannot include graphics of type: ⟨ext⟩` [graphics/graphicx]
You will get this error if you have specified a graphics type in the second argument of `\DeclareGraphicsRule` or used the `type` keyword of `\includegraphics` for which the loaded graphics driver has no support.

`\caption outside float`
A `\caption` command was found outside a float environment, such as a `figure` or `table`. This error message is disabled by some of the extension packages described in Chapter 6.

`Command` ⟨*name*⟩ `already defined`
You try to declare a command, an environment, a new savebox, a length, or a counter with a ⟨*name*⟩ that already has a meaning in LaTeX. Your declaration is ignored and you have to choose a different name. This error is also triggered if you use `\newcommand` with a ⟨*name*⟩ starting in `\end...`, even if `\renewcommand` claims the ⟨*name*⟩ is unused. It will also be issued if you try to define an environment ⟨*name*⟩ but the command `\end`⟨*name*⟩ already has a definition. For instance, you cannot define an environment `graf` because TeX has a low-level command called `\endgraf`.

`Command` ⟨*name*⟩ `invalid in math mode`
This is either a warning or an error message indicating that you have used a command in math mode that should be used only in normal text. In case of an error message, use `h` to get further help.

`Command` ⟨*name*⟩ `not defined as a math alphabet`
This error is issued when you try to use `\SetMathAlphabet` on a ⟨*name*⟩ that was not previously declared with `\DeclareMathAlphabet` or `\DeclareSymbolFontAlphabet` to be a math alphabet identifier.

`Corrupted NFSS tables`
LaTeX tried some font substitution and detected an inconsistency in its internal tables. This error happens if font substitution was triggered and the substitution rules contain a loop (i.e., some circular `sub` declarations exist) or when the default substitution arguments for the current encoding point to a nonexistent font shape group.

`Counter too large`
This error is produced if you try to display a counter value with `\fnsymbol`, `\alph`, or `\Alph` and the value is outside the available range for the chosen display form.

`Dimension too large` TeX
TeX can only deal with absolute sizes that are less than `16383.99998pt` (about 226 inches). Even on a huge page this range should be enough.

`\displaybreak cannot be applied here` amsmath
An enclosing environment such as `split`, `aligned`, or `gathered` has created an unbreakable block.

`Division by 0` graphics/graphicx
Usually, you will get this error when you scale a graphic that has a height of zero. This can happen unintentionally—for example, if you specify

`angle=-90,height=3cm` on `\includegraphics`. The rotation turns the image sideways, making the height zero, a value difficult to scale. In such a case use `totalheight` instead.

`Double subscript` TEX

Two subscripts appear in a row (e.g., `x_i_2`) and LaTeX does not know whether you mean x_{i2} or x_{i_2}. Add braces to indicate the subscripts: `x_{i_2}`.

`Double superscript` TEX

LaTeX found two superscripts in a row. See the explanation above.

`Encoding file '`⟨*name*⟩`' not found` fontenc

If you ask for encoding ⟨*enc*⟩, LaTeX tries to load the definitions for this encoding from the file ⟨*enc*⟩`enc.def` (after converting ⟨*enc*⟩ to lowercase letters). If this encoding file does not exist or cannot be found by LaTeX, you will get this error message.

`Encoding scheme` ⟨*name*⟩ `unknown`

The encoding scheme ⟨*name*⟩ you have specified in a declaration or in `\fontencoding` is not known to the system. Either you forgot to declare it using `\DeclareFontEncoding` or you misspelled its name.

`Environment` ⟨*name*⟩ `undefined`

You get this error if you use `\renewenvironment` on an environment name that is unknown to LaTeX. Either the ⟨*name*⟩ was misspelled or you should have used `\newenvironment` instead.

`Erroneous nesting of equation structures;` amsmath
`trying to recover with 'aligned'`

Only certain amsmath display structures can be nested; `aligned` is one of these, so the system replaces a wrongly nested environment with it. This is probably not what you intended, so you should change the wrongly nested environment.

`Extra & on this line` amsmath

This error occurs only when you are using old amsmath environments that are not described in this book. If it does occur, then it is disastrous and you need to check very carefully the environment where it occurred.

`Extra alignment tab has been changed to \cr` TEX

If you use an alignment structure, such as `tabular` or one of the display math environments (e.g., `eqnarray` or `split` from the amsmath package), then each row is divided into a defined number of columns separated by `&` signs. The error means that there are too many such characters, probably because you forgot a `\\` indicating the end of the row (`\cr` is TeX's name for the row end, but it is not a fully functional equivalent to `\\`).

`Extra \endgroup` TEX

TeX has seen an `\endgroup` without a preceding matching `\begingroup`.

`Extra \or` TeX
: TeX encountered an `\or` primitive that has no matching low-level `\ifcase` conditional. The extra `\or` can be the result from a bad use of `\ifthenelse`.

`Extra \right` TeX
: This error is issued by TeX if it finds a `\right` command without a matching `\left` in a formula. Recall that `\left`/`\right` pairs must be part of the same "sub-formula". They cannot, for example, be separated by `&` in an alignment or appear on different grouping levels.

`Extra }, or forgotten $` TeX
: This error is triggered when math formula delimiters (e.g., `$...$`, `\[...\]`) and brace groups are not properly nested. TeX thinks it has found a superfluous `}`, as in `$x}$`, and is going to ignore it. While in this example the deletion of the closing brace is the right choice, it would be wrong in `\mbox{\(a}`. There a closing `\)` is missing, so deleting the `}` will produce additional errors.

`Extra }, or forgotten \endgroup` TeX
: The current group was started with `\begingroup` (used, for example, by `\begin{..}`) but TeX found a closing `}` instead of the corresponding `\endgroup`. You will get this error if you leave a stray `}` inside a body of an environment.

`File '`⟨*name*⟩`' not found`
: LaTeX is trying to load the file ⟨*name*⟩ but cannot find it, either because it does not exist or because the underlying TeX program is looking in the wrong place. If the file exists but LaTeX claims it is not available, it is possible that your TeX installation uses a hashing mechanism to speed up file access, and you may have to run a special program to make your installation aware of newly installed files (e.g., mktexlsr with the TeX Live distribution on the CD-ROM).

  The error is issued by commands like `\input` and `\usepackage` if they cannot find the requested file. You can suggest an alternate file in response to the error. If the new name is specified without an extension, the old extension is reused if known to LaTeX. If you want to omit loading the file, press ⟨*Enter*⟩; to quit the run, type `x` or `X`. In some cases you might receive a similar low-level TeX error "! I can't find file '⟨*name*⟩'" that is slightly more difficult to quit; see the entry on page 901.

  If a graphics file requested with `\includegraphics` is missing, it may help to press `h` to learn which extensions have been tried when looking for the file.

`File ended while scanning` ⟨*something*⟩ TeX
: This error is part of a "`Runaway...`" error; check the explanations on page 909.

`Float(s) lost`
: One or more floats (e.g., `figure` or `table`) or `\marginpar` commands have not been typeset. The most likely reason is that you placed a float environment or marginal note inside a box by mistake—inside another float or `\marginpar`,

or inside a `minipage` environment, a `\parbox`, or a `\footnote`. LaTeX might detect this problem very late, such as when finishing the document. This can make it very difficult to find the offending place in the source. The best solution in this case is to half your document repeatedly (for example, by using the primitive `\endinput`), until the fraction producing the error is small enough that you spot it.

If incorrect nesting is not the root cause, then you may have encountered a serious coding problem in the float algorithm, probably caused by some extra packages you loaded.

`Font family ⟨cdp⟩+⟨family⟩ unknown`

You tried to declare a font shape group with `\DeclareFontShape` without first declaring the font ⟨*family*⟩ as being available in the encoding ⟨*cdp*⟩ using `\DeclareFontFamily`.

`Font ⟨name⟩ not found`

LaTeX's internal font tables contain wrong information, so LaTeX was unable to find the external font ⟨*name*⟩. Either this font was never installed, its `.tfm` file cannot be found by TeX for some reason, or the `\DeclareFontShape` declaration referring to it contains a spelling error.

`Font ⟨internal-name⟩=⟨external-name⟩ not loadable: ⟨reason⟩` TEX

TeX was unable to load a font with the LaTeX name ⟨*internal-name*⟩ having the structure `\`⟨*encoding*⟩`/`⟨*family*⟩`/`⟨*series*⟩`/`⟨*shape*⟩`/`⟨*size*⟩ in NFSS notation.[1] For example, it might say `\T1/cmr/m/it/10` (Computer Modern medium italic 10 points in T1 encoding). This should give you a good hint as to which font has a problem, even if you are not able to do much about it. There are two possible ⟨*reason*⟩s:

`Bad metric (TFM) file` TEX

The TeX metric file for the font (i.e., ⟨*external-name*⟩`.tfm`) is corrupted. Your installation may have some utility programs to check `.tfm` files in detail, although this usually requires expert help.

`Metric (TFM) file not found` TEX

The TeX metric file for the font (i.e., ⟨*external-name*⟩`.tfm`) was not found. Your installation may have a package (e.g., cmbright) to support a certain font family but the corresponding fonts are not available or are not properly installed.

`Font ⟨internal-name⟩=⟨external⟩ not loaded: Not enough room left` TEX

TeX can load only a certain number of fonts and there was no space left to load ⟨*internal-name*⟩. To find out which fonts are loaded, use the package tracefnt described in Section 7.5.6. One possible reason for excessive loading of fonts is the use of unusual font sizes for which LaTeX has to calculate and load the corresponding math fonts; see Section 7.10.7 for details.

[1]This is, in fact, a single command name, but due to the slashes in the name you cannot enter it directly in your document.

`Font shape` ⟨*font shape*⟩ `not found`

This error message is issued when there is something very wrong with a `\DeclareFontShape` declaration—perhaps if it does not contain any size specifications. Check the set-up for the font shape group in question.

`I can't find file '`⟨*name*⟩`'` TEX

A low-level TeX error raised when TeX cannot find a file that was requested to load. This error can be bypassed only by providing TeX with a file that it can find, or by stopping the run altogether (if your operating system allows that). To get past this error, many installations offer a file `null.tex` so that you can reply `null` in response. LaTeX normally uses the error message "File '⟨*name*⟩' not found", which supports various user actions. However, depending on the package coding, you may get the current error instead.

`I can't write on file '`⟨*name*⟩`'` TEX

TeX is not allowed to write data to the file ⟨*name*⟩. It is probably read-only or you may not have writing permission for its directory. On some TeX implementations (e.g., those on the TeX Live CD), the error may be preceded by a line like the following:

```
tex: Not writing to /texmf/tex/latex/base/latex.ltx (openout_any = p).
```

These TeX installations are by default configured to be "paranoid" (hence, "p" above) when writing to files. They allow you to write only to files below the current directory and *not* to any files specified with an absolute path name or starting with a dot in their name. To change that behavior you have to modify the settings in the file `texmf.cnf`.

`Illegal character in array arg`

You will get this error if the column specification for a `tabular` or `array` environment or a `\multicolumn` command contains characters that are not defined as column specifiers to LaTeX. A likely cause is that you used the extended syntax of the array package, described in Chapter 5, but forgot to load the package in the preamble (e.g., after you have copied a table from one document to another).

`Illegal parameter number in definition of` ⟨*command*⟩ TEX

This error occurs when a (re)defined command or environment uses `#`⟨*digit*⟩ in the replacement text, with a digit higher than the declared number of parameters. This error can be implicitly caused by nesting declaration commands, such as `\newcommand`, and forgetting that inner commands refer to their arguments by doubling the # characters; see page 846 for details. Another possible cause is referring to environment arguments in the second mandatory argument of `\newenvironment` or `\renewenvironment`.

`Illegal unit of measure (pt inserted)` TEX

You will get this error if you misspell or forget the unit when specifying the value for a length parameter; see Section A.1.5.

`Improper argument for math accent:` amsmath
`Extra braces must be added to prevent wrong output`

The whole of the "accented sub-formula" must be surrounded by braces.

`Improper discretionary list` TEX

This error is produced by TeX if it encounters a `\discretionary` command whose arguments contain anything other than characters, boxes, or kerns, after expansion.

`Improper \hyphenation` TEX

If you want to specify a hyphenation exception with `\hyphenation`, then you have to ensure that the argument contains only letters and - characters to indicate the hyphenation points. The problem is that, for example, accented characters in some font encodings are individual glyphs (allowed) but in other font encodings produce complicated constructs requiring the `\accent` primitive. For example, if the T1 encoding is used, then `\"u` refers to a single glyph. Thus,

```
\usepackage[T1]{fontenc} \hyphenation{T\"ur-stop-per}
```

is valid. The same hyphenation exception used with the default OT1 encoding would produce this error. See page 455 for an explanation of character differences in the major encodings.

`Improper \prevdepth` TEX

You used `\the\prevdepth` or `\showthe\prevdepth` outside of vertical mode, which is not allowed. This error will also show up if you mistakenly placed a float (e.g., a `figure` or `table`) inside a math display environment.

`Improper \spacefactor` TEX

You used `\the\spacefactor` or `\showthe\spacefactor` outside of horizontal mode, which is not allowed.

`\include cannot be nested`

LaTeX encountered an `\include` command inside a file loaded with `\include`. Because of implementation constraints this is impossible. Either change the inner `\include` into `\input` or rearrange your document file structure so that all `\include` statements are in the main document file.

`Incompatible list can't be unboxed` TEX

TeX was asked to unpack a box with horizontal material while trying to build a vertical list, or vice versa. Either you encountered a serious programming error in a package or you used some commands in a way explicitly not supported. For example, the commands from the soul package will produce this error when they are nested into each other.

`Incomplete` ⟨*conditional*⟩`; all text was ignored after line` ⟨*number*⟩ TEX

A low-level TeX conditional was unfinished (no matching `\fi`) when LaTeX reached the end of the current input file.

`Infinite glue shrinkage found` ⟨*somewhere*⟩ TeX

To break paragraphs into lines or the galley into pages, TeX assumes that there is no rubber length that can arbitrarily shrink, since that would mean that any amount of material can be placed into a single line or onto a single page. Thus, `\hspace{0pt minus 1fil}` in a paragraph, or `\vspace{0pt minus 1fil}` between paragraphs is not allowed and will raise this error (⟨*somewhere*⟩ gives some indication about where the offending material was found).

`Interruption` TeX

You will get this error after interrupting the LaTeX run (with Control-C or whatever your installation offers), so you should not be surprised by it. To finish the run prematurely, press `x` followed by ⟨*Return*⟩. Just pressing ⟨*Return*⟩ will continue the run.

`Invalid use of` ⟨*command*⟩ amsmath

You have used an amsmath command in a place where it does not make sense. Look up the correct use of this command.

`Keyboard character used is undefined in input encoding` ⟨*name*⟩ inputenc

The 8-bit number encountered in the document is not mapped by the input encoding ⟨*name*⟩ to some LICR object (see Sections 7.5.2 and 7.11.3). Check whether the document is really stored in the specified encoding.

`Language definition file` ⟨*language*⟩`.ldf not found` babel

When LaTeX processes the option list for babel and encounters an unknown option ⟨*language*⟩, it tries to load a file by the name of ⟨*language*⟩.ldf. This message is displayed when LaTeX fails to find it. This error can be caused by a simple typing mistake, or the file might not be stored on LaTeX's search path.

`Limit controls must follow a math operator` TeX

You can use `\limits` or `\nolimits` only following math operators such as `\sum`. See Table 8.4 for a list of common operator commands.

`\LoadClass in package file`

The `\LoadClass` command is only allowed in class files; see Section A.4.

`Lonely \item—perhaps a missing list environment`

The `\item` command is only allowed within list structures but LaTeX believes that this one was found outside a list.[1]

`Math alphabet identifier` ⟨*id*⟩ `is undefined in math version` ⟨*name*⟩

The math alphabet identifier ⟨*id*⟩ was used in a math version (⟨*name*⟩) for which it was not set up. An additional `\SetMathAlphabet` declaration should be added to the preamble of the document to assign a font shape group for this alphabet identifier.

[1]In contrast to the "...perhaps a missing `\item`" error, LaTeX's diagnosis in this case is usually correct.

`Math version ⟨name⟩ is not defined`

A math alphabet or a symbol font was assigned to a math version that is unknown to LaTeX. Either you misspelled its name or you forgot to declare this version (perhaps you have to add some package file). It is also possible that the math version you selected with `\mathversion` is not known to the system.

`Misplaced alignment tab character &` TeX

LaTeX found an `&` character outside of `tabular`, `align`, or one of the other alignment environments. If you want to typeset &, use `\&` instead. A possible cause is use of the amsmath environment `cases` or `matrix` without loading the package.

`Misplaced \cr` or `Misplaced \crcr` TeX

A `\cr` is the TeX low-level command for ending a row in an alignment structure (`\crcr` is a variation thereof); the corresponding LaTeX command is `\\`. TeX believes it came across such a command outside of an alignment structure.

`Misplaced \noalign` TeX

The TeX primitive `\noalign` is internally used to place "nonaligned" material between rows of alignment displays. It is therefore allowed only directly following the command that finishes a row. For example, you get this error when you use `\hline` outside of `array` or `tabular`, or not directly after `\\` within these environments.

`Misplaced \omit` TeX

The TeX primitive `\omit` is internally used to change the column specifications in an alignment display (e.g., to span rows with `\multicolumn` inside a `tabular`). The `\omit` command (and thus the commands calling it) is allowed only at the very beginning of an alignment cell (i.e., following `\\` or `&`).

`Missing \begin{document}`

This error occurs if typesetting is attempted while still within the document preamble.[1] It is most likely due to a declaration error that is misinterpreted by LaTeX. The error is also produced by text following `\begin{filecontents}` on the same line.

`Missing control sequence inserted` TeX

You used `\newcommand` or `\renewcommand` without providing a command name (starting with a backslash) as the first argument.

`Missing \cr inserted` TeX

TeX thinks it is about time to end the row in an alignment structure and inserted its low-level command for this purpose. In a LaTeX document, this guess is usually wrong, so TeX's recovery attempt usually fails in such a case.

[1]Typesetting inside an `\sbox` or `\savebox` declaration is accepted, but it is usually wise to move such declarations after `\begin{document}`, since some packages may delay their final set-up until that point.

`Missing delimiter (. inserted)` [TeX]
: A `\left`, `\right`, or one of the `\big..` commands was not followed by a delimiter. As corrective action the empty delimiter "`.`" was inserted. See Section 8.5.3 on page 498 for details.

`Missing \endcsname inserted` [TeX]
: This error can arise from using commands as part of the name of a counter or environment (e.g., `\newenvironment{Bl\"ode}`).

`Missing number, treated as zero` [TeX]
: This error occurs when TeX is looking for a number or a dimension but finds something else. For example, using `\value{page}` instead of `\thepage` would produce this error, since an isolated `\value` makes TeX expect a low-level counter assignment. In general, using a length register without a proper mutator function like `\setlength` can trigger this error. You also get this message when `\usebox` is not followed by a box bin defined with `\newsavebox`, since internally such bins are represented by numbers.

`Missing p-arg in array arg`
: There is a `p` column specifier not followed by an expression in braces (containing the width) in the argument to `tabular`, `array`, or `\multicolumn`.

`Missing @-exp in array arg`
: There is an `@` column specifier not followed by an expression in braces (containing the inter-column material) in the argument to `tabular`, `array`, or `\multicolumn`.

`Missing # inserted in alignment preamble` [TeX]
: An alignment preamble specifies the layout of the columns in an alignment structure. Internally, TeX uses # to denote the part of the column that should receive input. In LaTeX this is unlikely to appear as a first error.

`Missing = inserted for \ifnum` [TeX]
: TeX complains that the low-level `\ifnum` conditional is not followed by two numbers separated by <, =, or >. This error can occur when you forget the comparison operator in `\ifthenelse`.

`Missing = inserted for \ifdim` [TeX]
: The low-level `\ifdim` conditional is not followed by a comparison between two lengths.

`Missing $ inserted` [TeX]
: TeX has encountered something in normal text that is allowed only in math mode (e.g., `\sum`, `\alpha`, `^`), or something that is not allowed inside math (e.g., `\par`) while processing a formula. It has therefore inserted a `$` to switch to math mode or to leave it. If, for example, you tried to get an underscore by simply using `_` instead of `\_`, LaTeX would typeset the rest of the paragraph as a formula, most likely producing more errors along the way.

`Missing \endgroup inserted` TeX

This error indicates that a grouping structure in the document is incorrectly nested. Environments internally use `\begingroup` and `\endgroup` and for some reason TeX thinks that such a group was not properly closed. If you cannot determine why the group structure is faulty, try using the `\showgroups` or `\tracinggroups` feature of eTeX, as explained on page 917.

`Missing \right. inserted` TeX

Your formula contains a `\left` without a matching `\right`. Recall that `\left`/`\right` delimiter pairs must be part of the same "sub-formula"; they cannot, for example, be separated by & in an alignment or appear on different grouping levels.

`Missing { inserted` TeX

TeX thinks there is an open brace missing and inserted one. This error is, for example, caused by a stray } inside a `tabular` cell.

`Missing } inserted` TeX

Something is wrong in the grouping structure of the document and TeX tries to recover by inserting a closing brace. This attempt either gets it onto the right track again or causes you to receive more errors. Usually, the problem becomes apparent if you look at the typeset output. If you cannot determine why the group structure is faulty, try using the `\showgroups` or `\tracinggroups` feature of eTeX, as explained on page 917.

`Multiple \label's: label` ⟨*label*⟩ `will be lost` amsmath

Within the amsmath display environments, you can have only one `\label` per equation. It is usually best to remove all but the last, as it is the only one that will be effective.

`Multiple \tag` amsmath

Within the amsmath display environments, you can have only one `\tag` command per equation. All but the first will be ignored.

`No counter '`⟨*name*⟩`' defined`

The counter ⟨*name*⟩ referenced in either `\setcounter`, `\addtocounter`, or the optional argument of `\newcounter` or `\newtheorem` is unknown to LaTeX. It must first be declared with `\newcounter`.

`No Cyrillic encoding definition files were found` babel

The language definition files for the supported "Cyrillic languages" check whether any of the known Cyrillic font encoding files (e.g., T2A, T2B) can be found. If not, this error message is displayed and you need to install Cyrillic support for LaTeX first.

`No declaration for shape` ⟨*font shape*⟩

The sub or ssub size function used in a `\DeclareFontShape` command refers to a substitution shape that is unknown to LaTeX's font selection scheme.

`No driver specified` color/graphics/graphicx

The package **graphics**, **graphicx**, or **color** was loaded without specifying a target device option. On most installations this is done using the configuration files `graphics.cfg` and `color.cfg`.

`No room for a new` ⟨*register*⟩ TeX

The packages loaded in your document require more internal registers (`\count`, `\dimen`, ...) than there are available in TeX. Try processing your document with eTeX and additionally load the **etex** package.

`No \title given`

A LaTeX class has executed `\maketitle` without seeing a `\title` declaration. Only `\date` is optional when this command is used.

`Not a letter` TeX

You specified a hyphenation exception with `\hyphenation` but the argument to this command contained some characters that TeX does not consider to be letters. For example, `\hyphenation{la-ryn-gol-o-gist's}` would produce such an error since `'` is not a "letter" in TeX's categorization.

`Not in outer par mode`

This error is issued when a `\marginpar` or a float environment, such as `table` or `figure`, encountered inside a box-producing command or environment. For instance, you cannot use a `\marginpar` in a footnote, a float, a `tabular`, or a similar place (since all of them produce boxes). Move the offending object to the main galley.

`Number too big` TeX

You assigned or used a number in `\setcounter` or `\addtocounter` that is larger than the largest number that TeX can handle (2147483647, hexadecimal 7FFFFFFF). This error can also happen when modifying a length register with `\setlength` or `\addtolength`.

`OK` TeX

You used a TeX tracing command, like `\show` or `\showthe`; after displaying the data LaTeX stopped with this message to allow for some interaction on the command line (e.g., entering `i\show..` to view some other values). This message is also shown if `\tracingonline` is positive and commands are used that normally only write to the transcript file; see the next message.

`OK (see the transcript file)` TeX

You used a TeX tracing command, like `\showbox` or `\showlists`, without also directing LaTeX to display the result on the terminal.

`Old form` ⟨*command*⟩ `should be \begin{`⟨*envname*⟩`}` amsmath

You have used `cases`, `matrix`, or `pmatrix` in its non-**amsmath** command form (probably with its old internal syntax). Change to the **amsmath** environment form with standard internal syntax.

`Only one # is allowed per tab` TEX
: This error indicates a broken alignment template. In LaTeX it should not occur, unless caused by a fragile command in a moving argument.

`Option clash for package` ⟨*name*⟩
: The package ⟨*name*⟩ was requested twice with a conflicting set of options. When you press `H` in response to this error, LaTeX will show you the sets of conflicting options. As LaTeX loads a package only once,[1] the best solution is to specify all options on the first occasion. If this is not possible, because the package is already loaded as part of the class or another package, you can try to specify the required options as global options to the `\documentclass` command. In an emergency you can even load a package before `\documentclass` by using `\RequirePackage`. See Section 2.1.1 for details.

`Page height already too large`
: You used `\enlargethispage` on a page whose vertical size is already larger than `8191.99998pt`, or roughly 113 inches. LaTeX thinks that this is dangerously large and will not extend the page size as requested.

`Paragraph ended before` ⟨*command*⟩ `was complete` TEX
: As discussed in Section A.1.2, commands defined with `\newcommand*` or `\renewcommand*` are not allowed to contain `\par` or an empty line. If they do, you will get a `Runaway argument` together with this error. The ⟨*command*⟩ listed may not be the one used in your document. For example, `\emph{..\par..}` will list `\text@command` in the error message (i.e., the internal command called by `\emph`).

`(Please type a command or say '\end')` TEX
: You have replied with ⟨*Return*⟩ in response to `*`. See first entry on page 894.

`\pushtabs and \poptabs don't match`
: You issued a `\poptabs` command in a `tabbing` environment, but there was no previous `\pushtabs` command issued.

`\RequirePackage or \LoadClass in Options Section`
: A `\RequirePackage` or `\LoadClass` was found inside a package or class file between the `\DeclareOption` commands and `\ProcessOptions`. Loading packages or classes in this part is not allowed as it would clobber the data structure holding the current set of options; see Section A.4 for details. If you want to load a package when a certain option is specified, use a flag to indicate that the option was selected and load it after the `\ProcessOptions` command has done its job.

`Rotation not supported` graphics/graphicx
: You have requested rotation with `\rotatebox` or a similar command but the selected graphics driver does not support rotation of objects. LaTeX will leave

[1] The only exception is the fontenc package, which can be loaded as often as needed with different options; see Section 7.5.3 on page 361.

the right amount of space but the printed document might show the image in the wrong position.

`Runaway` ⟨*something*⟩ TEX

TEX thinks it has scanned too far while looking for the end of ⟨*something*⟩, where ⟨*something*⟩ can be either `argument`, `definition`, `preamble`, or `text`. Unless low-level TEX code is at fault, the most likely cause is `argument`. For example, you forgot the closing brace of an argument, it might cause TEX to scan until it reaches the end of the file or until its memory is filled—whichever comes first. Incomplete definitions done with `\newcommand`, `\newenvironment`, and so forth also claim that the `argument` has run away. Only low-level definitions, involving TEX primitives like `\def`, produce a `Runaway definition`.

A `Runaway preamble` means that an alignment structure has problems (that should not occur in normal LATEX documents) and `Runaway text` usually refers to a token register assignment (this should never happen unless there is a serious package implementation error).

In contrast to the situation with normal error messages, you will not get a line number that indicates where the error was detected (since TEX often has reached the end of the file). Instead, you will see the beginning of the material that was being absorbed. For example, if you have a definition without the final closing brace,

```
\newcommand\foo{bar
\begin{document} Some text \end{document}
```

you will get

```
Runaway argument?
{bar \begin {document} Some text \end {document}
! File ended while scanning use of \@argdef.
<inserted text>
                \par
<*> samplefile.tex

?
```

The fact that TEX in that case inserted `\par` as a recovery action is of little help, since the complete document was already swallowed. Instead of "`File ended while...`", you might see some other message at this point, such as "`Paragraph ended before...`".

`Scaling not supported` graphics/graphicx

You have requested scaling with `\resizebox` or a similar command but the selected graphics driver does not support scaling of objects. LATEX will leave the right amount of space but the printed document will show the image at the original (unscaled) size.

`Something's wrong-perhaps a missing \item`

This error message is produced by an `\addvspace` command when encountered in horizontal mode. The follow-up remark about "`perhaps a missing`

\item" is unfortunately seldom correct. For example, forgetting the closing brace on \mbox as in \mbox{...\section{..}... would produce this error, since the \section command that executes \addvspace internally is now used in horizontal mode.

Identify which command issued the \addvspace causing the error, and check whether that command was used incorrectly. Refer to page 858 for an in-depth discussion of the \addvspace command.

Sorry, I can't find ⟨*format*⟩ ... [TeX]

If you get this message, then LaTeX never started because TeX did not find the ⟨*format*⟩ containing the basic LaTeX definitions. There is a problem with your TeX installation and you have to consult the installation documentation.

Suggested extra height (⟨*value*⟩) dangerously large

Using the ⟨*value*⟩ with \enlargethispage would make the resulting page too large (more than 113 inches) for LaTeX's liking.

Symbol font ⟨*name*⟩ is not defined

You tried to make use of the symbol font ⟨*name*⟩—for example, within a \DeclareMathSymbol command—without declaring it first with a \DeclareSymbolFont declaration.

Symbol ⟨*command*⟩ not provided by font family ⟨*name*⟩ [textcomp]

The **textcomp** package implements the TS1 encoding, which is unfortunately implemented fully by just a minority of the font families usable with LaTeX. The package will typeset the symbol using a default family stored in \textcompsubstdefault. You can turn the error into a warning by loading **textcomp** with the option warn. See Section 7.5.4 for more details.

Tab overflow

LaTeX supports up to 13 tabulator positions (\=) inside a tabbing environment, and you have used a larger number. If not all of them are needed at the same time, you can try solving the problem by using \pushtabs and/or providing template lines with \kill.

\tag not allowed here [amsmath]

The \tag command is allowed only within the top level of a mathematical display. It is usually best to move it to the end of the logical equation in which it occurs.

TeX capacity exceeded, ⟨*explanation*⟩ [TeX]

TeX ran out of some sort of memory and died. This error is discussed in detail in Section B.1.1 on page 915.

Text line contains an invalid character [TeX]

The input file contains a strange, nonprinting character that is rejected by TeX. This may happen if you used a word processor to create the file and did not save it as "text".

`The attribute` ⟨*attrib*⟩ `is unknown for language` ⟨*lang*⟩ babel
You tried to activate an attribute for a language ⟨*lang*⟩ that is not defined in the language definition file for this language. Check the documentation of babel with respect to this language.

`The character '`⟨*char*⟩`' is not a shorthand character in` ⟨*language*⟩ babel
When a user uses the command `\shorthandon` and passes it a ⟨*char*⟩ that is not defined to be a shorthand for the current ⟨*language*⟩, this error message is displayed and the instruction is ignored.

`The font size command \normalsize is not defined...`
A class file needs to provide a minimal set-up, including a definition for `\normalsize`; see Section A.4.9 on page 888 for details.

`There's no line here to end`
This error is triggered if `\newline` or `\\` is found outside a paragraph (i.e., after a `\par` or an empty line). If the intention was to produce extra vertical space, use `\vspace` or any of the other commands described on page 857.

`This may be a LaTeX bug`
To the author's knowledge, until now this message never actually signaled a LaTeX bug. It means, however, that LaTeX got thoroughly confused by previous errors and lost track of the state of its float data structure. It is best to stop and correct previous errors first.

`This NFSS system isn't set up properly`
This error occurs when LaTeX detects a mistake while trying to verify the font substitution tables at `\begin{document}`. It means that either a `\DeclareFontSubstitution` or `\DeclareErrorFont`[1] declaration is corrupted. These declarations need to point to valid font shapes (declared with `\DeclareFontShape`). Type `h` for additional information and inform your system maintainer. If you are the system maintainer, read the end of Section 7.10.5.

`Too deeply nested`
Standard LaTeX supports a total of six levels of lists nested in each other. Those levels can include up to four lists of type `itemize` or `enumerate`. This error signals that your document has overflowed one of these limits. You probably have forgotten to end some list environments properly. If you really need additional levels, you need to copy the base definitions for `list`, `itemize`, and/or `enumerate` into a private package and modify their hard-wired constants.

`Too many columns in eqnarray environment`
The `eqnarray` environment supports a maximum of three columns (i.e., two & signs per row). For serious math, consider the amsmath package described in Chapter 8, which allows for more complex display structures.

[1]The declaration `\DeclareErrorFont` is used during installation and points to a font (font shape + size) that should be used when everything else fails. Its default is Computer Modern Roman 10pt, which should be available with any TeX installation. See [109] for further details.

`Too many math alphabets used in version` ⟨*name*⟩

You used too many different math alphabet identifiers in your formulas. If this error occurs after adding the bm package, define `\newcommand\bmmax{0}` before loading bm and try again; this prevents the package from preallocating math alphabets.

`Too many unprocessed floats`

Floats that cannot be placed immediately are deferred by LaTeX, possibly causing subsequent floats to be deferred as well. LaTeX can defer up to 18 floats, then you will receive this error message. Using the package morefloats will increase this limit to 36 but if there is a float that cannot be placed for some reason this change will merely delay receiving the above error. See Chapter 6 for ways to deal with this situation.

This error can also be triggered if you have too many `\marginpar` commands within a single paragraph. A `\marginpar` temporarily uses two storage bins for deferred floats as long as the current paragraph has not been typeset (this allows a maximum of nine marginal notes per paragraph, or fewer if there are already some deferred floats).

`Two \documentclass or \documentstyle commands`

Only one such command is allowed per document. Your document includes more than one, perhaps as the result of combining two originally separate documents.

`Two \LoadClass commands`

A class can load at most one other class to do the bulk of processing. See Section A.4 for a detailed discussion of how classes are built.

`Undefined color` ⟨*name*⟩ color

You have requested a color with `\color` or a similar command from the color package without previously defining it with `\definecolor`. See [57] or the color package documentation for details.

`Undefined control sequence` TeX

This is perhaps the most common of all LaTeX errors, though it shows up as a TeX error message: you have used a command name that was not previously defined. Often you may have simply mistyped the name in your document (e.g., `\bmox` instead of `\mbox`). To carry on in such a case, you can respond with `i\mbox`, inserting the correct name. Later on you can correct your source document. It is also possible to get this error as a result of using a fragile command in a moving argument.

`Undefined font size function` ⟨*name*⟩

A size function used in `\DeclareFontShape` was misspelled. Check the entry or tell your system maintainer.

`Undefined tab position`

This error is raised if you try to advance in a `tabbing` environment with `\>`, `\+`, `\-`, or `\<` to a tabulator position that was not previously set up with `\=`.

Either the `\=` is actually missing or perhaps you have used `\+` or `\pushtabs` and got confused when specifying the tabular position to which you actually want to move.

`Unknown graphics extension:` ⟨*ext*⟩ graphics/graphicx
: You will get this error if you try to load a fully specified graphics file (with extension ⟨*ext*⟩) and the graphics driver does not know the particular extension and there is no default rule set up. The `dvips` program, for example, interprets every unknown extension as EPS, so with this driver you will never see this error but probably others.

`Unknown option '`⟨*option*⟩`' for package '`⟨*name*⟩`'`
: You specified an ⟨*option*⟩ for package ⟨*name*⟩ that is not declared by that package. Consult the package documentation on the available options.

`Use of` ⟨*command*⟩ `doesn't match its definition` TeX
: Low-level macro definitions made with `\def`, instead of `\newcommand` and friends, sometimes require special argument delimiters (e.g., the `(..)` of the picture commands). If ⟨*command*⟩ is a LaTeX command, check its syntax. Otherwise, this is most likely a spurious error due to using a fragile command in a moving argument without `\protect`.

`\usepackage before \documentclass`
: The `\usepackage` declaration can be used only after the main class was loaded with `\documentclass`. Inside a class file you instead have to use `\RequirePackage`.[1]

`UTF-8 string \u8:`⟨*8-bit-sequence*⟩ `not set up for LaTeX use` inputenc
: The Unicode character denoted by the UTF-8 ⟨*8-bit-sequence*⟩ is not known to LaTeX. Under the precondition that it is available in a font encoding used in the document, it has to be set up using the `\DeclareUnicodeCharacter` declaration; see Section 7.11.3 on page 443.

`\verb ended by end of line`
: To better detect errors, the argument of `\verb` must be placed on a single line. Thus, this error signals that you either forgot the final delimiter for the argument or the argument was broken over several lines in the source. In case of very long arguments, it may help to split them over several `\verb` commands and, if necessary, masking a line break in the source with a `%` sign.

`\verb illegal in command argument`
: Except in very special situations (explicitly documented in this book), it is not possible to use `\verb` (or `verbatim`) in the argument of other commands. If you need verbatim text in such a place, use, for example, `\SaveVerb` and `\UseVerb` from the **fancyvrb** package described in Section 3.4.3.

`You already have nine parameters` TeX
: LaTeX supports command or environment definitions with a maximum of

[1] It is technically possible to load a package before a class by using `\RequirePackage`, but this should be avoided unless you know what you are doing.

nine parameters, but your \newcommand or \newenvironment specified 10 or more.

`You can't use 'macro parameter #' in ⟨some⟩ mode` TEX
: TEX found a stray # character somewhere that does not seem to be a reference to an argument of some command. If you wanted to typeset this symbol, use \# instead.

`You can't use '\spacefactor' in vertical mode` TEX
: TEX lets you refer to the \spacefactor only when you are building a horizontal list. You will get this error when you use the LaTeX command \@ outside of a paragraph. Since many internal commands start with an @ in their names, you might also get this error if you use code containing such internal commands (e.g., \@startsection) in the preamble of your document without surrounding it with \makeatletter and \makeatother. In that case TEX sees \@ followed by the letters startsection, and a later use of this code then executes \@ that in turn produces this error message.

`You can't use '\prevdepth' in horizontal mode` TEX
: The \prevdepth dimension can be used only while in vertical mode (i.e., between paragraphs).

`You can't use '\end' in internal vertical mode` TEX
: This is one of the more misleading TEX error messages, since it refers to the TEX primitive \end (ending a TEX run) that was redefined by LaTeX to become the end-tag of environments. The error means that LaTeX's \end{document} or the \stop command was encountered while LaTeX was building a box. For example, \begin{figure}...\stop would generate it.

`You can't use '⟨command⟩' in ⟨some⟩ mode` TEX
: TEX complains that ⟨*command*⟩ is not allowed in one of its modes. Some specific variations of this theme have already been discussed. If you haven't used ⟨*command*⟩ directly, then the most likely cause for this error is a broken fragile command in a moving argument.

`You haven't defined output directory for '⟨path⟩'` docstrip
: The configuration file docstrip.cfg contains a declaration for \BaseDirectory but the internal ⟨*path*⟩ in the DOCSTRIP script has no translation to a local directory. Use \DeclareDirectory or \UseTDS in docstrip.cfg to specify a translation as described in Section 14.2.3 on page 830.

`You haven't defined the language ⟨language⟩ yet` babel
: Various user interface commands of babel check whether their argument is a language that was specified in the option list when babel was loaded. If the ⟨*language*⟩ was *not* specified, processing is stopped and this error message is displayed.

`You haven't specified a language option` [babel]

This message is shown when no known languages have been specified for babel—that is, neither in the option list to babel nor in the global option list (this is likely to be due to a typo). You should expect that processing your document will nevertheless produce many more errors.

B.1.1 Dying with memory exceeded

The TeX program contains a number of internal tables of fixed size used for storing away different kinds of information needed at run time. Whenever any of these tables overflows, LaTeX will abort with a "`TeX capacity exceeded`" error.

Until the mid-1990s, memory problems could, in fact, be due to the size of the document. In some cases it was impossible to process a document as a whole.[1] These days such limitations are gone or are at least less severe. For one, the average TeX implementation is already equipped with huge internal tables. In addition, most implementations allow you to modify the table sizes via configuration files instead of requiring you to manually recompile TeX. In some cases you may have to generate a new LaTeX format; for more details, consult the documentation of your TeX distribution.[2]

Nevertheless, people experience this dreadful error once in a while, usually as the result of a faulty command definition. Below are four candidates reduced to the bare bones of the problem we want to discuss—in reality, such problems usually lurk in more complex definitions.

```
\newcommand\FAILa{.\FAILa}            \newcommand\FAILb{\FAILb x}
\newcommand\FAILc{\typeout{.}\FAILc} \newcommand\FAILd{.\par\FAILd}
```

If you execute `\FAILa` as defined above, you will receive the following output (the reported memory size possibly differs) after a short while:

```
! TeX capacity exceeded, sorry [main memory size=1500001].
\FAILa ->.
          \FAILa
```

The `main memory` is the part of TeX in which macro definitions and the material for the current page are stored. Looking at the above recursive definition, it is clear that it generates a never-ending sequence of periods. Since paragraph breaking is deferred until TeX sees a `\par` command or a blank line to globally optimize the line breaks, TeX waits in vain for a chance to break the paragraph material into lines.

[1]The first edition of this book required a specially compiled version of the TeX program with all such tables enlarged by a factor of 10 and could be processed only on a large UN*X workstation.

[2]The TeX live distribution, which comes with this book, lets you specify the size of most tables through the configuration file `texmf.cnf`. See the TeX live manual for details.

Exceeding main memory because of too many macro definitions is less likely these days. Nevertheless, even that can happen (in theory) if the size of this memory is small and you load many packages, have a large number of huge deferred floats, or use macro packages[1] that produce new macros on the fly.

If you get this error only with larger documents and LaTeX actually produces pages before giving up, you can try to find out whether the memory is gradually filling up (which suggests a table size problem) by setting `\tracingstats=2` in the preamble of your document. TeX will then report the main memory status after finishing each page, producing output like the following:

```
[765]
Memory usage before: 4262&161788; after: 1286&157691; still untouched: 1323176
[766]
Memory usage before: 3825&160983; after: 636&156520; still untouched: 1323176
[767]
Memory usage before: 3652&160222; after: 771&156307; still untouched: 1323176
```

The number reported to the left of the `&` is the memory devoted to large objects such as boxes; the number on the right is the amount of memory used by macro definitions and character data. Thus, one can expect a reduction in both values whenever a page has finished (i.e., the `after:` value). If the right-hand value is slowly increasing, however, then something is probably adding more and more definitions.

If we use `\FAILb`, we overflow a different table. Here the recursion happens before LaTeX actually reaches the end on the macro expansion and thus needs to store away the unprocessed part of the expansion.

```
! TeX capacity exceeded, sorry [input stack size=1500].
\FAILb ->\FAILb
                x
```

With today's size for the `input stack`, this message usually appears only if a recursion like the one above makes that stack grow at a frightening speed. In a normal LaTeX document you will seldom find nested definitions that make this stack grow beyond a value of 50 (for this book the maximum value was 35).

What happens if you execute either `\FAILc` or `\FAILd`? Both are similar to `\FAILa` but neither overflows any internal TeX table. Instead, both will simply fill your hard disk. The only action of `\FAILc` is to show periods on your screen and in the transcript file, thereby very slowly filling up the disk with a huge transcript. `\FAILd`, on the other hand, contains a `\par` in its definition and therefore is able to typeset paragraphs (each consisting of a single dot); as a result it produces pages in rapid succession. Such an experiment ended on the author's machine with a document containing 22279 pages and the following message:

```
tex: fwrite: No space left on device
```

[1]For example, varioref defines two labels internally for every use of `\vref`, which can result in a noticeable amount of memory consumption in large documents.

On your private machine, this is merely a nuisance, easily rectified. On systems with shared resources, however, you should be careful when letting LaTeX run unattended. This type of error once hit a student very badly; this individual processed such a document on a mainframe in batch mode without a time or size limit and was presented a bill for computer processing time of several thousand dollars.

Several other internal tables can overflow in principle. Below is the complete list of those not already discussed, along with an explanation for the most likely reason for the overflow. Some additional information can be found in [82, p. 300].

`buffer size` The characters in the lines being read from a file. Since the default size is usually quite large, the most likely cause for an overflow is lost line breaks due to a faulty conversion of a file during transfer from one operating system to another. A buffer overflow can also be caused by some PC word processing programs, which internally put an entire paragraph on a single line even though the text appears to be broken into several lines on the screen.

`exception dictionary` The number of hyphenation exceptions as specified by `\hyphenation`. LaTeX has some exceptions specified for the English language, and some language packages specify additional exceptions. However, if this table overflows, you must have been doing a very thorough job.

`font memory` The font metric data loaded by LaTeX. These days an overflow is unlikely. If it happens, LaTeX has loaded too many fonts—probably because you used many different font sizes and LaTeX calculated and loaded math fonts for all the sizes. Increase the table size, if possible, or refer to Chapter 7 for information on how to reduce the number of fonts.

`grouping levels` The number of unfinished groups that delimit the scope for setting parameters, definitions, and other items—for instance, braces, the start of environments, or math mode delimiters. An overflow usually indicates a programming error (e.g., a definition that opens more groups than it closes). That type of error is sometimes difficult to identify. Good help is available with the eTeX program,[1] which offers the command `\showgroups` to produce a listing of stacked groups starting with the innermost one. For example, placing it into the footnote on the current page will yield

```
### semi simple group (level 3) entered at line 2955 (\begingroup)
### insert group (level 2) entered at line 2955 (\insert0{)
### semi simple group (level 1) entered at line 2921 (\begingroup)
### bottom level
```

The semi simple group on level 1 is due to the fact that this text is typeset in a `description` environment (the `\begin` command issues internally a `\begingroup` command). The `\footnote` command is implemented with the TeX primitive `\insert`, which contributes level 2. In fact, another semi simple group is started by `\footnote`, which ensures that color changes remain local.

[1] In modern distributions LaTeX is automatically using the eTeX program. On older installations you may have to call a different program (e.g., `elatex` instead of `latex`) when processing a document.

What we can deduce from this example is that the relationships among top-level document commands and internal groups are far from obvious or simple. However, the line numbers that show when a group was entered do help, since there are usually no long-ranging groups in normal documents.

As an alternative, the eTeX program offers the internal tracing counter `\tracinggroups`. If it is set to a positive number, the entry and exit of groups is recorded in the transcript file; with `\tracingonline` having a positive value, this information also appears on screen.

`hash size` The number of command names known to TeX. Most packages contribute a fixed number of new command names. Each `\label` or `\bibitem` command in the document generates one new internal command name. Thus, packages that internally use the `\label` command (e.g., varioref) may significantly contribute to filling that table in large documents.

`number of strings` The number of strings—command names, file names, and built-in error messages—remembered by TeX. In some cases TeX is able to free unused space but usually such strings survive even if they are used only locally. One possible reason for overflowing this table is the use of many files in an application. Each opening for reading or writing of a file contributes, even when the same file is used many times over.

For historical reasons, TeX has a somewhat unusual string-handling concept involving several tables, each of which can overflow. Thus, if you change the `hash size` to allow for more commands, you may need to adjust the `number of strings` and quite likely the `pool size`, and vice versa.

`parameter stack size` The total number of command parameters of nested commands being expanded but not yet fully processed. For example, suppose a command with 4 arguments calls a command with 5 arguments, which in turn calls a command with 3 arguments, thereby using up 12 slots in this table. The moment TeX reaches the end of a macro replacement text it will free the stack. Thus, with today's implementations it is quite difficult to hit that limit, unless you use a flaky recursive definition with arguments, for example:

```
\newcommand\FAIL[3]{\typeout{Got #1, #2 and #3 but \FAIL is a mess}\DO}
```

Do you see the problem? Since the `\typeout` contains `\FAIL` by mistake, it gets called again, before its replacement text has been fully processed (picking up the characters `i`, `s`, and `a` as arguments). As a result, `\DO` is never executed and we finally get

```
! TeX capacity exceeded, sorry [parameter stack size=1500].
\FAIL #1#2#3->
              \typeout {Got #1, #2 and #3 but \FAIL is a mess}\DO
l.18 \FAIL 123
```

This is similar to the \FAILb example from page 916, except that because of the number of arguments the `parameter stack` overflowed first.

`pattern memory` The memory available to store hyphenation patterns. This table cannot overflow during normal document processing, since such patterns are loaded only during format generation. If you receive this error during that process, reduce the number of languages for which you load hyphenation patterns into your format. These days pattern loading is normally defined in the file `language.dat`.

`pool size` The characters in strings—command names and file names (including the full path on some implementations). If this table overflows, the most likely cause is the use of too many files, especially if they have long absolute path names. This can, for example, happen if a document includes many graphics and one uses `\graphicspath` to make LaTeX search for the images in several directories—every attempt to open a file contributes to this string pool.

`save size` The set of values to restore when a group ends. With today's default limits, this is again difficult to overflow. The most likely cause is the use of both local and global assignments to the same object, something that can happen only through the use of low-level TeX programming, since LaTeX assignments are either always local (for most types) or always global (e.g., counter assignments).

To avoid unnecessary growth of the `save stack`, the `document` environment has a special implementation[1] so that it does not produce a group (as normal environments do). Without it every new definition would automatically push an unnecessary "undefined" value onto the `save stack`—unnecessary, because by the time that group would end all processing would stop anyhow.

`semantic nest size` The number of token lists being worked on simultaneously. Boxes, math formulas, and other elements start a new list, suspending work on the current structure. Once they are finished TeX has to continue constructing the suspended object, so all such unfinished objects are remembered in the `semantic nest` stack. With a default size of several hundred objects, it is very difficult to get even close to this limit with normal documents.[2] In an emergency, TeX offers `\showlists`, which displays all unfinished lists that TeX is currently working on.

`text input levels` The number of simultaneously open input sources (e.g., files opened by `\include`, `\input`, or `\usepackage`). On the author's implementation of TeX one would need to nest 1500 files to reach this limit.

[1] As a side effect it is impossible to use `\begin{document}` inside another environment since the grouping structure is not obeyed.

[2] The author could not think of any problematic definition that would not hit any of the other limits first.

B.2 Warnings and informational messages

While error messages make LaTeX stop and wait for user input, warning messages are simply displayed on the terminal and in the transcript file and processing continues. If applicable, LaTeX also shows the source line number that triggered the warning. The warnings are prefixed by "`LaTeX Warning:`" or "`LaTeX Font Warning:`" if they are issued by the core LaTeX code. Otherwise, they identify the issuing package or class by starting with "`Package` ⟨*name*⟩ `Warning:`" or "`Class` ⟨*name*⟩ `Warning:`", respectively. TeX warnings, such as "`Overfull...`", have no standard prefix string.

In addition to warnings, LaTeX writes informational messages to the transcript file without displaying this information on the terminal. To better distinguish between informational and warning messages, warnings are shown in blue in the following alphabetical listing.

`Calculating math sizes for size` ⟨*text size*⟩
: LaTeX has to guess the correct font sizes for subscripts and superscripts because it could not find the information for the current ⟨*text size*⟩ in its internal tables. This message usually is followed by several font size correction warnings because LaTeX's initial guess is seldom successful. This situation can arise when you select an uncommon size using the `\fontsize` command; see Section 7.10.7 if the math formulas look strange.

`Checking defaults for` ⟨*cdp*⟩/⟨*font shape*⟩
: This message is written in the transcript file at `\begin{document}` while LaTeX is verifying that the substitution defaults for the encoding ⟨*cdp*⟩ are sensible. It is followed either by `... okay` or by an error message that is generated when the ⟨*font shape*⟩ group specified with `\DeclareFontEncoding` is unknown to LaTeX.

`Citation '`⟨*key*⟩`' on page` ⟨*number*⟩ `undefined`
: The ⟨*key*⟩ specified as an argument to `\cite` or `\nocite` is not defined by a `\bibitem` command or you need another run of LaTeX (and perhaps BIBTeX) to make it known to LaTeX. The latter case is indicated by an additional warning, "Label(s) may have changed...", as discussed on page 924. The page number is omitted if the warning is emitted by `\nocite`.

`Command` ⟨*name*⟩ `invalid in math mode`
: This is either a warning or an error message indicating that you have used a command in math mode that should be used only in normal text. A warning will be generated when an obsolete, yet still valid, construction is used.

`Document Class:` ⟨*name*⟩ ⟨*date*⟩ ⟨*additional-info*⟩
: This line is produced by a `\ProvidesClass` command in the document class code. Although not a warning, it appears both on the terminal and in the transcript file. If a document produces different output on different installa-

tions, you should compare the "`Document Class:`", "`File:`", and "`Package:`" messages to identify any release differences.

`Empty 'thebibliography' environment`
This warning is issued if a `thebibliography` environment has no `\bibitem` commands. It often indicates a problem with a BibTeX run. For example, the BibTeX program may have been unable to resolve a single citation.

`Encoding` ⟨*name*⟩ `has changed to` ⟨*new name*⟩ `for ...`
This warning is issued when in the declaration of a symbol font different encoding schemes in different math versions have been used. It may mean that the `\DeclareMathSymbol` commands for this symbol font are not valid in all math versions.

`(\end occurred` ⟨*when*⟩ TeX
You receive this warning at the very end of your run whenever TeX finds the `\end{document}` or `\stop` command to be premature. As a warning the message is unfortunately misleading, because it refers to a TeX primitive `\end` that was reused by LaTeX to become the environment end-tag. The ⟨*when*⟩ can be one of two cases:

`inside a group at level` ⟨*number*⟩`)` TeX
In this case the LaTeX run ended while there were still some open groups. Such groups include explicit braces that are not closed (e.g., `{\itshape..`), use of `\bgroup` and `\begingroup` in macro code without their counterparts, and unclosed environments in the source. The latter normally triggers a suitable LaTeX error first (i.e., "`\begin{`⟨*env*⟩`}` on...") unless you ended the run with `\stop`, since in that case no check for mismatched environments is made.

`when` ⟨*condition*⟩ `on line` ⟨*line number*⟩ `was incomplete)` TeX
In this case LaTeX completed the run while a low-level TeX conditional remained unfinished. With LaTeX documents using only standard commands, this problem should not occur unless you ended the document inside a file loaded with `\include`. In other cases it probably means there is a bug in a package. Try to identify the source of the conditional (by looking at the ⟨*line number*⟩) to see in which command it was used. Note that the ⟨*line number*⟩ may not be in the current file—unfortunately, TeX does not divulge the file name. In very difficult situations you can try to use eTeX's advanced tracing options to pinpoint the problem: if `\tracingifs` is set to 1, you will get detailed trace information about nested conditionals as they are executed.

`External font` ⟨*name*⟩ `loaded for size` ⟨*size*⟩
LaTeX has ignored your request to load some font shape at size ⟨*size*⟩ and has loaded the external font ⟨*name*⟩ instead. (This message is generated by the size function `fixed`.)

`Faking` ⟨*command*⟩ `for font family` ⟨*name*⟩ `in TS1 encoding` textcomp
The glyph ⟨*command*⟩ is not available in the `TS1` encoding of the current font family. LaTeX has responded by "faking" it in some way. This is, for example, done for the `\texteuro` glyph (€), if unavailable. Section 7.8.7 describes ways to get a real euro symbol.

`File '`⟨*name*⟩`' already exists on the system.`
`Not generating it from this source`
This warning is generated by a `filecontents` environment when the file ⟨*name*⟩ already exists somewhere in the search path of LaTeX. If you want to unpack the file nevertheless, either delete (or rename) the version found by LaTeX or extract the file manually with the help of an editor.

`File:` ⟨*name*⟩ ⟨*date*⟩ ⟨*additional-info*⟩
This line is produced from the `\ProvidesFile` command used to identify a file and its last modification date. By convention, the ⟨*additional-info*⟩ starts with a version number, though it is not required. Although of the same importance as `\ProvidesClass`, this information is written only to the transcript file to avoid cluttering the terminal with messages. If a document produces different output on different installations, you should compare the "`Document Class:`", "`File:`", and "`Package:`" messages to identify any release differences.

`File:` ⟨*encoding*⟩⟨*family*⟩`.fd` ⟨*date*⟩ ⟨*additional-info*⟩
This important special case of the previous informational message indicates that a font definition file for some ⟨*encoding*⟩ (usually displayed in lowercase) and ⟨*family*⟩ combination was loaded. Such files contain font shape group declarations and are described in Section 7.10.6.

`Float too large for page by` ⟨*value*⟩
A float is too tall by ⟨*value*⟩ to fit in the current `\textheight`. It will be printed on a page by itself (if permitted), thereby possibly overflowing into the bottom margin. If the float is not allowed to go on a float page, it will prevent all further floats in its class from being placed.

`Font shape` ⟨*font shape*⟩ `in size` ⟨*size*⟩ `not available`
LaTeX issues this message when it tries to select a font for which the requested font attribute combination is not available and a substitution is defined in the internal tables. Depending on the contents of these tables, one of the following additional messages will be issued:

`external font` ⟨*name*⟩ `used`
LaTeX has selected the external font ⟨*name*⟩ in that particular situation and does not know to which font shape group it belongs. (This message is generated by the size function `subf`.)

`size` ⟨*size*⟩ `substituted`
LaTeX has selected the correct shape, but since the requested size is not

available LaTeX has chosen the nearby size ⟨*size*⟩. This action is taken automatically if none of the simple sizes or size ranges in the ⟨*font shape*⟩ group declaration matches.

`shape` ⟨*font shape*⟩ `tried`

LaTeX has selected a different ⟨*font shape*⟩ group because the requested one is not available for the requested ⟨*size*⟩. (This message is generated by the size function sub.)

`Font shape` ⟨*font shape*⟩ `undefined. Using` '⟨*other shape*⟩' `instead`

This warning is given when a combination of font attributes is specified for which LaTeX has no font shape definition. For example, requesting `\fontseries{b}\ttfamily` would normally trigger this warning, since Computer Modern fonts have neither bold typewriter nor bold extended typewriter. However, when the latter combination is requested, you will not receive this warning but only some information in the transcript file because for `\textbf{\texttt{..}}` the `.fd` files contain an explicit substitution rule.

If LaTeX identifies a particular symbol that it cannot typeset in the requested shape, the above warning is followed by "`for symbol` ⟨*name*⟩".

`Font shape` ⟨*font shape*⟩ `will be scaled to size` ⟨*size*⟩

LaTeX has loaded the requested font by scaling it to the desired size. To print a document containing scaled fonts, your printer driver must have these fonts in the correct size or must be able to scale them automatically.

`Foreign command` ⟨*command*⟩`;` amsmath
`\frac or \genfrac should be used instead`

Although the use of ⟨*command*⟩ is not an error, you are strongly discouraged from using this old form for your (generalized) fractions in LaTeX. Use the amsmath commands instead.

`Form feed has been converted to Blank Line`

The `filecontents` environment detected a "form feed" character (`^^L`) in the source and will write it as an empty line (`\par` command if interpreted by LaTeX) into the external file. As `filecontents` was designed to distribute textual data, it cannot be used for handling arbitrary binary files.

`'h' float specifier changed to 'ht'` or
`'!h' float specifier changed to '!ht'`

You specified `h` or `!h` as a float placement without giving any other options. LaTeX requires some alternative in case "here" leads to an impossible placement because not enough room is left on the current page. If you really want to prevent floats from floating, consider using the float package described in Section 6.3.1.

`Ignoring text` '⟨*text*⟩' `after \end{`⟨*env*⟩`}`

This warning is issued by `filecontents` or `filecontents*` when textual material is detected following the `\end` tag.

`Label '⟨key⟩' multiply defined`

The document contains two or more `\label` commands with the same ⟨*key*⟩. References to this ⟨*key*⟩ will always refer to the last `\label` defined. Ensure that all ⟨*key*⟩s are different.

`Label(s) may have changed. Rerun to get cross-references right`

LaTeX has detected that the label definitions, as compared to those in the previous run, have been modified and that (at least) one additional LaTeX run is necessary to resolve cross-references properly.

In theory it is possible, though unlikely, that this message will persist regardless of the number of processing runs.[1] If this is the case, compare the `.aux` files of different runs to determine which label alternates between different states and resolve the problem manually.

`Loose \hbox (badness ⟨number⟩) ⟨somewhere⟩` TeX

TeX produced a horizontal box with a badness of 13 or greater (which corresponds to using 50% or more of the available stretchability). This warning can be safely ignored unless you are a perfectionist; in fact, it will not be produced unless you change the default for `\hbadness`. See the message "Underfull `\hbox`..." on page 928 for more details.

`Loose \vbox (badness ⟨number⟩) ⟨somewhere⟩` TeX

TeX produced a vertical box with a badness of 13 or greater (which corresponds to using 50% or more of the available stretchability). The warning is produced only if `\vbadness` was set to a value below 100. See the message "Underfull `\vbox`..." on page 930 for more details.

`Making ⟨char⟩ an active character` babel

For each character that is turned into a shorthand character, this information message will be written to the transcript file. When a document shows unexpected results, this information might help if the problems are caused by inadvertent use of a shorthand character.

`Marginpar on page ⟨number⟩ moved`

A `\marginpar` could not be aligned with the text line to which it was originally attached, because a preceeding `\marginpar` already occupies the space.

`Missing character: There is no ⟨char⟩ in font ⟨name⟩!` TeX

Search out for this message in the transcript

Although this message usually indicates a serious problem, unfortunately it is only written to the transcript file (unless `\tracingonline` is positive). It means that somewhere in the source a request for a symbol ⟨*char*⟩ was made for which the current font (⟨*name*⟩ is the external name) has no glyph in the corresponding position. The displayed ⟨*char*⟩ may differ on different TeX

[1] For example, if the `\label` is near the page boundary between pages "iii" and "iv", the use of `\pageref` before the `\label` might result in a situation where the `\label` will be moved to page "iv" if the textual reference "iii" is used, and vice versa.

installations.[1] For example, using the command `\symbol` can produce this warning because you can ask for any font slot with this command. However, standard font-encoding-specific commands, as discussed in Section 7.11.4 on page 455, should never produce this warning.

`No \author given`
You used `\maketitle` without specifying an author first. In contrast to a missing `\title` this omission generates a warning.

`No auxiliary output files`
This information is displayed when you use a `\nofiles` declaration in the document preamble.

`No characters defined by input encoding change to` ⟨*name*⟩
The input encoding file ⟨*name*⟩`.def` does not seem to contain any input encoding declarations. For the `ascii` encoding, this is the expected behavior; for all other encodings, it indicates a problem.

`No file` ⟨*name*⟩
LaTeX displays this information whenever it tries to read from an auxiliary file (e.g., `.aux` or `.toc`) but cannot find the file. This is not considered an error since such files are created only after the first run. However, the same routine is also used by `\include`, so that, unfortunately, a missing "include file" will trigger this unsuspicious warning too..

`No hyphenation patterns were loaded for the language` `'`⟨*language*⟩`'` babel
All language definition files check whether hyphenation patterns for the language selected were loaded into the LaTeX format. If this is not the case, this message is displayed and a default set of hyphenation patterns will be used. The default patterns are those loaded into pattern register 0 (typically American English).

`No input encoding specified for` ⟨*language*⟩ `language` babel
This message can appear when no specific input encoding was specified in the document and one of the supported languages needs the Cyrillic alphabet for typesetting. For these languages several input encodings are popular; therefore, the language definition insists that the one used *must* be explicitly mentioned.

`No positions in optional float specifier. Default added ...`
A float environment (e.g., `figure` or `table`) was used with its optional placement argument, but it did not contain any suitable information. Hence, LaTeX used its default placement rules.

[1] Sometimes you see something like `^^G`, sometimes real characters are displayed. Unfortunately, there is no guarantee that they correspond to your input: some translation that depends on the operating system may happen when the characters are written to the transcript file.

`Oldstyle digits unavailable for family` ⟨*name*⟩ [textcomp]
You used `\oldstylenums` with a font family that does not contain old-style digits. As an emergency measure LaTeX produced lining digits (from the current font family) instead. See Section 7.5.4 for details.

`Optional argument of \twocolumn too tall on page` ⟨*number*⟩
The material in the optional argument to `\twocolumn` was so tall, that fewer than three lines remain on the page. LaTeX will not start two-column mode on the current page and will start a new page instead.

`\oval, \circle, or \line size unavailable`
The requested size for the mentioned commands is unavailable. LaTeX will choose the closest available size. See, for example, Section 10.4.3 for ways to avoid this problem.

`Overfull \hbox (`⟨*number*⟩`pt too wide)` ⟨*somewhere*⟩ [TeX]
TeX was forced to build a horizontal box (e.g., the line of a paragraph or a `\makebox`) of a certain width and was unable to squeeze the material into the given width, even after shrinking any available space as much as possible. As a result, the material will stick out to the right. In most cases this is quite noticeable, even if the total amount is small. You have to correct this problem manually, since TeX was unable to resolve it (Sections 3.1.11 and B.3.3 give some advice). For a list and explanation of the possible origins (i.e., the ⟨*somewhere*⟩), see the warning "Underfull `\hbox`..." on page 928.

`Overfull \vbox (`⟨*number*⟩`pt too wide))` ⟨*somewhere*⟩ [TeX]
TeX was asked to build a vertical box of a fixed size (e.g., a `\parbox` or a `minipage` with a second optional argument; see Appendix A.2.2 on page 866) and found more material than it could squeeze in. The excess material will stick out at the bottom. Whether this result poses a problem depends on the circumstances. For a list and explanation of the possible origins (i.e., the ⟨*somewhere*⟩), see the warning "Underfull `\vbox`..." on page 930.

`Overwriting encoding scheme` ⟨*something*⟩ `defaults`
This warning is issued by `\DeclareFontEncodingDefaults` when it overwrites previously declared defaults for "text" or "math".

`Overwriting` ⟨*something*⟩ `in version` ⟨*name*⟩ `...`
A declaration, such as `\SetSymbolFont` or `\DeclareMathAlphabet`, changed the assignment of font shapes to ⟨*something*⟩ (a symbol font or a math alphabet) in math version ⟨*name*⟩.

`Package:` ⟨*name*⟩ ⟨*date*⟩ ⟨*additional-info*⟩
This line is produced by the `\ProvidesPackage` command, which is used to identify a package and its last modification date. By convention, the ⟨*additional-info*⟩ starts with a version number, though it is not required. Although of the same importance as `\ProvidesClass`, this information is written to just the transcript file to avoid cluttering the terminal with messages. If

a document produces different output on different installations, you should compare the "`Document Class:`", "`File:`", and "`Package:`" messages to identify any release differences.

`Redeclaring font encoding` ⟨*name*⟩
This warning is issued if `\DeclareFontEncoding` is used for an encoding that is already defined (thereby potentially changing its defaults).

`Redeclaring math accent` ⟨*name*⟩
This warning is issued if `\DeclareMathAccent` is used for a math accent that was previously declared. If the command to be declared is known but not an accent, you get an error message instead.

`Redeclaring math alphabet` ⟨*name*⟩
A `\DeclareMathAlphabet` or `\DeclareSymbolFontAlphabet` command was issued to declare ⟨*name*⟩, which was already defined to be a math alphabet identifier. The new declaration overrides all previous settings for ⟨*name*⟩.

`Redeclaring math symbol` ⟨*name*⟩
The command ⟨*name*⟩ was already declared as a math symbol and your declaration overrides the old definition.

`Redeclaring math version` ⟨*name*⟩
You issued a `\DeclareMathVersion` command for a version that was already declared. The new declaration overrides all previous settings for this version with the default values.

`Redeclaring symbol font` ⟨*name*⟩
You issued a `\DeclareSymbolFont` command for a symbol font that was previously declared. The new declaration overrides the symbol font in all known math versions.

`Reference '`⟨*key*⟩`' on page` ⟨*number*⟩ `undefined`
A reference created with `\ref`, `\pageref`, or one of the other cross-reference commands discussed in Chapter 2 used a ⟨*key*⟩ for which LaTeX has not seen a corresponding `\label` command. If the `\label` is somewhere in the document, you simply need another LaTeX run to make it known to LaTeX. This situation is indicated by the additional warning "Label(s) may have changed..." discussed on page 924.

`Size substitutions with differences up to` ⟨*size*⟩ `have occurred`
This message will appear at the end of the run if LaTeX selected at least one significantly different font size because a requested size was not available. The ⟨*size*⟩ is the maximum deviation that was needed.

`Some font shapes were not available, defaults substituted`
This message will appear at the end of the run if LaTeX had to use automatic font substitution for some font shapes.

`Tab has been converted to Blank Space`

The `filecontents` environment detected a "tab" character (`^^I`) in the source and will write it as a space into the external file.

`Text page` ⟨*number*⟩ `contains only floats`

One or more floats processed as "top" or "bottom" floats are together so tall that very little space (less than two lines) is left for normal text on the current page. Therefore, LaTeX decided to place only floats on the page in question (even if some or all of the floats do not explicitly allow for this placement). This message can appear only when the placement parameters for floats were changed drastically from their default values; see the beginning of Chapter 6 for details.

`There were multiply-defined labels`

This warning appears at the end of a LaTeX run when LaTeX detected at least one pair of `\label` or `\bibitem` commands with the same key. Check the transcript file and make sure that all keys used are different.

`There were undefined references`

This warning appears at the end of a LaTeX run when LaTeX detected references to unknown keys and concluded that rerunning the document would not resolve them. You should check the transcript file for all occurrences of "Reference ⟨*key*⟩ undefined" and "Citation ⟨*key*⟩ undefined" and correct them, either by fixing a misprint or by adding the necessary `\label` or `\bibitem` commands. In case of missing citation ⟨*keys*⟩, all you may have to do is rerun BibTeX and then LaTeX.

`Tight \hbox (badness` ⟨*number*⟩`)` ⟨*somewhere*⟩ TeX

TeX produced a horizontal box and had to shrink the interior spaces. You will see this message only if `\hbadness` is set to a value less than `100`. See the message "Underfull `\hbox`..." below for more details.

`Tight \vbox (badness` ⟨*number*⟩`)` ⟨*somewhere*⟩ TeX

TeX produced a vertical box and had to shrink the interior spaces. You will see this message only if `\vbadness` is set to a value less than `100`. See the message "Underfull `\vbox`..." on page 930 for more details.

`Try loading font information for` ⟨*cdp*⟩+⟨*family*⟩

You will find such a message in the transcript file whenever LaTeX tries to load a `.fd` file for the encoding/family combination ⟨*cdp*⟩/⟨*family*⟩.

`Unable to redefine math accent` ⟨*accent*⟩ amsmath

This warning is rare but it may be issued when loading the amsmath package with nonstandard mathematical fonts.

`Underfull \hbox (badness` ⟨*number*⟩`)` ⟨*somewhere*⟩ TeX

TeX was forced to build a horizontal box (e.g., the line of a paragraph or a `\makebox`) of a certain width, and the white space within that box had to stretch more than it was designed to do (i.e., stretched more than 100% of the available `plus` parts in stretchable spaces). Internally, this situation is expressed by a badness value greater than 100; a value of 800 means that twice the total stretchability was used to produce the required width.[1]

Whether such an underfull box actually presents a noticeable problem is something that you may have to check visually in the produced output. If the badness is 10000 the box can be arbitrarily bad. Since TeX's value for infinity is quite low, it might mean that TeX has favored one very bad line over several bad but still acceptable lines that appear in succession. In that case using `\emergencystretch` can help you; see Section 3.1.11.

The limit of badness values above which such warnings are shown is controlled by the integer parameter `\hbadness`. LaTeX's default is 1000, so warnings appear only for really bad boxes. If you want to produce an important document try a more challenging value, such as `\hbadness=10`, to find out how many lines TeX really considers imperfect.

Note that the warning always talks about `\hbox`, regardless of the actual box construct used in the source, since it is directly generated by TeX. The location where the problem occurred is indicated by ⟨*somewhere*⟩, which is one of the following four possibilities:

`detected at line` ⟨*line number*⟩ TeX

An explicitly constructed box (construction ending at line ⟨*line number*⟩ in the source) has the problem—for example, a `\makebox` with an explicit width argument or some other LaTeX construct that builds boxes.

`has occurred while \output is active` TeX

TeX was in the process of building a page and encountered the problem while attaching running headers and footers and the like. Since this is an asynchronous operation, no line number is given. Look at the page generated closest to where the warning was issued to determine whether it warrants manual correction.

`in alignment at lines` ⟨*line numbers*⟩ TeX

The box is part of a `tabular` or some math alignment environment. The ⟨*line numbers*⟩ give you the source position of the whole alignment structure, since by the time TeX encounters the problem it no longer has a way to relate it back to the source in more detail.

`in paragraph at lines` ⟨*line numbers*⟩ TeX

The underfull box is due to a badly spaced line in the paragraph (source line numbers given as ⟨*line numbers*⟩). The additional symbolic display of the line in question should help you to pinpoint the problem.

[1]The exact formula is $\min(100r^3, 10000)$ where r is the ratio of "stretch used" to "stretch available", unless there is infinite stretch present (e.g., introduced by a command like `\hfill`), in which case the badness will be zero.

`Underfull \vbox (badness ⟨number⟩) ⟨somewhere⟩` TeX

TeX was forced to build a vertical box (e.g., a `\parbox` or a `minipage`) of a certain height, and the vertical space in that box had to stretch more than it was supposed to; see the discussion of badness and stretchability in the description of the "Underfull `\hbox`..." warning. You can suppress all warnings for badness values below a certain limit by setting `\vbadness=`⟨*value*⟩. Then LaTeX issues warnings only for boxes with a badness larger than ⟨*value*⟩ (the default is `1000`). The ⟨*somewhere*⟩ indicates the origin of the problem and can be one of the following cases:

`detected at line ⟨line number⟩` TeX

The box was explicitly constructed (the ⟨*line number*⟩ points to the end of the box construction) and there is not enough stretchable space available. For example,

```
\parbox[c][2in][s]{4cm}{test test}
```

would produce this warning because the box should be 2 inches high and the contents should fill this height (argument `[s]`), but there is nothing stretchable available. For instance, something like `\par\vfill` between the two words. See Appendix A.2.2 for details on paragraph boxes.

`has occurred while \output is active` TeX

In the most frequent case, the space on the current page needed stretching beyond acceptable limits in TeX's eyes. Whether this is visually a real problem depends on many factors, such as the type of spaces on the page. For example, a large stretch in front of a heading is usually less severe than a spaced-out list. Thus, the best advice is to check such pages manually. Often, `\enlargethispage` or `\pagebreak` will help.

If the problem appears surprisingly often, then the spacing parameters for lists, paragraphs, and headings should be examined to see whether they are too rigid (see Chapters 2 to 4). Also check whether the `\textheight` corresponds to an integral number of text lines; see the discussion on page 197.

`in alignment at lines ⟨line-numbers⟩` TeX

This warning should not arise with standard LaTeX but can occur in some specialized applications. In such a case use ⟨*line-numbers*⟩ to identify the source lines in your document.

`Unused global option(s): [⟨option-list⟩]`

Some of the options specified on `\documentclass` have been used by neither the class nor any package in the preamble. A likely reason is that the names of the options have been misspelled. Also note that some packages do not react to global options, but only to those explicitly specified when loading the package. See Appendix A.4 for details.

`Writing file '`⟨*name*⟩`'`

This informational message is produced by both `filecontents` and `filecontents*` when they write their body to an external file ⟨*name*⟩.

`Writing text '`⟨*text*⟩`' before \end{`⟨*env*⟩`} as last line of` ⟨*file*⟩

This warning is issued by the `filecontents` or `filecontents*` environment when it detects textual material directly preceeding the `\end` tag.

`You have more than once selected the attribute '`⟨*attrib*⟩`'` babel
`for language` ⟨*language*⟩

This message is displayed if the same attribute is entered more than once in the second argument of `\languageattribute`; only the first occurrence will trigger the activation of the attribute.

`You have requested` ⟨*package-or-class*⟩ `'`⟨*name*⟩`',`
`but the` ⟨*package-or-class*⟩ `provides '`⟨*alternate-name*⟩`'`

You requested loading of ⟨*name*⟩ via `\usepackage` or `\RequirePackage` (in case of a package) or via `\documentclass` or `\LoadClass` (in case of a class), but the package or class provides a variant of the original with the internal name ⟨*alternate-name*⟩. Unless this was a typo by the package or class provider, it means that your installation has a package or class variant that is likely to behave differently from the original. Thus, your document may be formatted differently when processed on another installation. Whether this is the correct behavior is something you need to investigate by looking at the package or class in question.

Specifying a relative or absolute path name triggers this warning as a side effect.

`You have requested release '`⟨*date*⟩`' of LaTeX,`
`but only release '`⟨*old-date*⟩`' is available`

A `\NeedsTeXFormat` command has requested a LaTeX release of at least ⟨*date*⟩ but the date of your format is ⟨*old-date*⟩. Usually, such a request is made to ensure that a certain feature of the LaTeX format is available, so it is likely that your document will produce additional errors or strange formatting later. Update to a more recent version of LaTeX.

`You have requested, on line` ⟨*num*⟩`, version '`⟨*date*⟩`' of` ⟨*name*⟩`,`
`but only version '`⟨*old-date*⟩`' is available`

A class or package was required to have a date not older than ⟨*date*⟩ but the version on your installation is from the date ⟨*old-date*⟩. Update the class or package in question.

B.3 TEX and LATEX commands for tracing

In this section we discuss tools and techniques for tracing and for displaying status information—for example, finding out why something is strangely spaced on the page or why your own command definition does the wrong thing.

B.3.1 Displaying command definitions and register values

In many situations it is useful to get some information about LaTeX's current internals, the precise definitions of commands, the values of registers, and so on. For example, if the use of \newcommand reports that the command to be defined is already defined, you may want to know its current definition, to ensure that you do not redefine an important command.

Displaying command definitions

For this purpose TeX offers the command \show, which displays the definition of the token following it and then stops and displays a question mark while waiting for user intervention. For example, after defining \xvec as in Example A-1-4 on page 844, we can display its definition as follows:

```
\newcommand\xvec[1]{\ensuremath{x_1,\ldots,x_{#1}}}
\show\xvec
```

This will produce the following output on the terminal and in the transcript file:

```
> \xvec=\long macro:
#1->\ensuremath {x_1,\ldots ,x_{#1}}.
l.6 \show\xvec

?
```

The first line, which starts with >, shows the token being displayed (\xvec) and gives its type (\long macro), indicating that \xvec is a macro that accepts \par commands in its argument; in other words, this macro was defined with \newcommand rather than \newcommand*. The second line shows the argument structure for the command (up to ->), revealing that the command has one argument (#1). Note that while the argument on the \newcommand declaration was indicated with [1], it is now shown differently. The rest of the line—and possibly further lines, if necessary—shows the definition part. The code is terminated with a period that is not part of the definition but helps to identify stray spaces at the end of the definition, if any. Note that the code display is normalized. Thus, after a command that would swallow subsequent spaces, you will see a space regardless of whether a space was coded in the original definition.

Following the display of the definition, the source line (including the line number in the input file) is shown. Then LaTeX stops with a question mark. To continue you can press enter. Alternatively, you can type h to see what other possibilities are available.

Not all commands produce such easily understandable output. Assume that you try to display a command that was defined to have an optional argument, such as \lvec as defined in Example A-1-5 on page 845:

```
\newcommand\lvec[2][n]
 {\ensuremath{#2_1+\cdots + #2_{#1}}}
\show\lvec
```

In that case you will get this result:

```
> \lvec=macro:
->\@protected@testopt \lvec \\lvec {n}.
```

Apparently, the `\lvec` command has no arguments whatsoever (they are picked up later in the processing). And something else is strange in this output: what is `\\lvec`? Is it the command `\\` followed by the letters `lvec`, or is it a strange command `\\lvec` that has two backslashes as part of its name? It is actually the latter, though there is no way to determine this fact from looking at the output of the `\show` command. Such strange command names, which cannot be generated easily by the user, are sometimes used by LATEX internally to produce new command names from existing ones using `\csname` and `\endcsname` and other low-level mechanisms of TEX.

Displaying internal commands with strange names

So what should you do, if you want to see the definition of `\\lvec`? It should be clear that writing `\show` in front of such a command will not work, as in normal situations TEX will see `\\` and think that it is the command to "show". For that reason, you have to use the same low-level mechanisms first to generate the command name in a way that it is considered a single token by TEX and then to feed this token to `\show`:

```
\expandafter\show\csname \string\lvec \endcsname
```

Technically, what happens is that a command name is generated from the tokens between `\csname` and `\endcsname`. Inside that construct, the `\string` command turns the command `\lvec` into a sequence of characters starting with a backslash that no longer denotes the start of a command. This is why the resulting command name contains two backslashes at the beginning. The `\expandafter` command delays the evaluation of the following `\show` command so that `\csname` can perform all of its work before `\show` is allowed to look at the result.

That's quite a mouthful of low-level TEX, but after typing it in, we are rewarded with the following output:

```
> \\lvec=\long macro:
[#1]#2->\ensuremath {#2_1+\cdots + #2_{#1}}.
<recently read> \\lvec
```

This time we do not see a source file line after the command display, but the words `<recently read>`. They indicate that TEX has assembled the token `\\lvec` somewhere in memory rather than reading it directly from a file.

What would happen if we forgot the initial `\expandafter` in the previous input? We would get the following result:

```
> \csname=\csname.
l.5 \show\csname
                \string\lvec \endcsname
?
! Extra \endcsname.
l.5 \show\csname \string\lvec \endcsname

?
```

First we are told that \csname is a \csname, which seems like totally useless information but, in fact, indicates that \csname is a primitive command or register already built into the TEX program—in contrast to, say, \lvec, which was a macro defined via \newcommand. LATEX also shows how far it has read the input line by placing the unread tokens (\string and friends) into the next line. Since we carry on, TEX will stop again shortly (after having consumed the whole line) to complain about a spurious \endcsname because the matching \csname was shown but not executed.

Detecting a primitive command

Displaying register values

The \show command is useful for learning about commands and their definitions or finding out if something is a primitive of TEX. But it does not help in finding the current values of length or counter registers. For example,

```
\show\parskip \show\topmargin \show\topsep
```

will give us the following information:

```
> \parskip=\parskip.
l.5 \show\parskip
                  \show\topmargin \show\topsep
?
> \topmargin=\dimen73.
l.5 \show\parskip \show\topmargin
                                  \show\topsep
?
> \topsep=\skip23.
l.5 \show\parskip \show\topmargin \show\topsep
```

From the above we can deduce that \parskip is a TEX primitive (the fact that it is a rubber length is not revealed), that \topmargin is actually the \dimen register (rigid length) with register number 73, and that \topsep is the \skip register (rubber length) with number 23.

If we want to know the value of any such register, we need to deploy a different TEX primitive, called \showthe instead of \show, which gives us the following output on the terminal and also proves that \parskip is, indeed, a rubber length:

```
> 0.0pt plus 1.0pt.
l.5 \showthe\parskip
```

Using \showthe in this way allows us to display the values of the length registers allocated with \newlength and of internal TEX registers such as \baselineskip and \tolerance. What we cannot display directly with it are the values of LATEX counters allocated with \newcounter. For this we have to additionally deploy a \value command that turns a LATEX counter name into a form that is accepted by \showthe. For example,

```
\showthe\value{footnote}
```

would show the current value of the footnote counter on the terminal.

Instead of displaying the meaning of a macro or the value of a register on the terminal, you can alternatively typeset this kind of data by using \meaning instead of \show and \the instead of \showthe. The output is slightly different: the name of the token is not shown by \meaning; instead, only its type and "meaning" is presented. Compare the next example with the output shown earlier in this section.

Typesetting command definitions or register values

B-3-1

```
\long macro:#1->\ensuremath
{x_1,\ldots ,x_{#1}}
0.0pt plus 1.0pt
16.0pt
8.0pt plus 2.0pt minus 4.0pt
footnote=0
```

```
\newcommand\xvec[1]{\ensuremath{x_1,\ldots,x_{#1}}}
\ttfamily    % use typewriter
\raggedright
\meaning\xvec    \par \the\parskip\par
\the\topmargin   \par \the\topsep \par
footnote=\the\value{footnote}
```

If displaying command definitions or register values is insufficient for determining a problem, you can alternatively trace the behavior of the commands in action; see Section B.3.5 on page 945.

B.3.2 Diagnosing page-breaking problems

Once in a while LATEX produces unexpected page breaks or shows some strange vertical spaces and you would like to understand where they are coming from or what precise dimensions are involved. For these tasks TEX offers a few low-level tracing tools.

Symbolic display of the page contents

If you specify \showoutput somewhere in your document, TEX will display (starting with the current page) symbolic representations of complete pages on the terminal and the transcript file. This will generate a large amount of output, of which we will show some extracts that have been produced by compiling the first paragraph of this section separately.

Every page output will start with the string Completed box being shipped out followed by the current page number in brackets. Then you get many lines showing the boxes that make up the page, starting with a \vbox (vertical box) and its sizes in pt containing the whole page. To indicate that something is the contents of a box, everything inside is recursively indented using periods instead of blanks. Spaces, even if they are rigid, are indicated by the keyword \glue (see line 3 or 6); stretchable space has some plus and/or minus components in its value, as we will see later. Whether it is a horizontal or a vertical space is determined by the box in which this space is placed. For example, the \glue of 16.0pt on line 3 is a vertical space that came from \topmargin; see also Example B-3-1. In the extract you also see an empty \vbox of height 12pt (lines 5 to 7), which is the empty running header, followed in line 8 by the space from \headsep (25pt), followed by the box containing the text area of the page starting at line 10. Lines 15 and following show how individual characters are displayed; here `\T1/cmr/m/n/10` indicates the font for each character. The `\glue` in between (e.g., line 19), marks an interword space with its stretch and shrink components.

```
1  Completed box being shipped out [1]
2  \vbox(633.0+0.0)x407.0
3  .\glue 16.0
4  .\vbox(617.0+0.0)x345.0, shifted 62.0
5  ..\vbox(12.0+0.0)x345.0, glue set 12.0fil
6  ...\glue 0.0 plus 1.0fil
7  ...\hbox(0.0+0.0)x345.0
8  ..\glue 25.0
9  ..\glue(\lineskip) 0.0
10 ..\vbox(550.0+0.0)x345.0, glue set 502.00241fil
11 ...\write-{}
12 ...\glue(\topskip) 3.1128
13 ...\hbox(6.8872+2.15225)x345.0, glue set - 0.17497
14 ....\hbox(0.0+0.0)x15.0
15 ....\T1/cmr/m/n/10 O
16 ....\T1/cmr/m/n/10 n
17 ....\T1/cmr/m/n/10 c
18 ....\T1/cmr/m/n/10 e
19 ....\glue 3.33252 plus 1.66626 minus 1.11084
20 ....\T1/cmr/m/n/10 i
21 ....\T1/cmr/m/n/10 n
22 ....\glue 3.33252 plus 1.66626 minus 1.11084
23 ....\T1/cmr/m/n/10 a
```

As a second example from a page trace, we show the symbolic display of the structures near a line break. You see the space added by TEX at the right end of a text line (`\rightskip` on line 5) and the box containing the line. Thus, line 6 is outdented again. It contains a symbolic representation for the costs to TEX to break after this line, indicated by the command `\penalty`. The actual value here is due to the value of the `\clubpenalty` parameter.[1] This is followed in line 7 by the vertical space added between the lines, computed by TEX by taking the value of `\baselineskip` and subtracting the depth of the previous line box and the height of the following line box, which starts at line 8.

```
1  ....\T1/cmr/m/n/10 s
2  ....\T1/cmr/m/n/10 o
3  ....\T1/cmr/m/n/10 m
4  ....\T1/cmr/m/n/10 e
5  ....\glue(\rightskip) 0.0
```

[1]The penalty to break after the first line in a paragraph is given by the integer parameter `\clubpenalty`; the cost for breaking before the last line by `\widowpenalty`. Both default to 150, that is, they slightly discourage a break.

```
...\penalty 150
...\glue(\baselineskip) 2.96054
...\hbox(6.8872+1.94397)x345.0, glue set 0.55421
....\T1/cmr/m/n/10 s
....\T1/cmr/m/n/10 t
```

As a final example, we look at some part of the symbolic page output produced from a line like this:

```
\begin{itemize} \item test \end{itemize}  \section{Test}
```

The particular part of interest is the one generated from `\end{itemize}` and `\section{Test}`. What we see here (lines 1 to 7) is a curious collection of `\glue` statements, most of which cancel each other, intermixed with a number of `\penalty` points:

```
1  ...\penalty -51
2  ...\glue 10.0 plus 3.0 minus 5.0
3  ...\glue -10.0 plus -3.0 minus -5.0
4  ...\penalty -300
5  ...\glue 10.0 plus 3.0 minus 5.0
6  ...\glue -10.0 plus -3.0 minus -5.0
7  ...\glue 15.0694 plus 4.30554 minus 0.86108
8  ...\glue(\parskip) 0.0 plus 1.0
9  ...\glue(\baselineskip) 8.12001
10 ...\hbox(9.87999+0.0)x345.0, glue set 290.70172fil
```

These lines are generated from various `\addpenalty` and `\addvspace` commands issued; for example, lines 1 and 2 are the penalty and the rubber space added by `\end{itemize}`. The `\section` command then adds a breakpoint to indicate that the place before the section is a good place to break a page (using `\@secpenalty` with a value of -300). In fact, the break should be taken before the `\glue` from line 2, or else there would be a strange space at the bottom of that page. As it is technically impossible to remove material from the vertical galley, `\addpenalty` uses the trick to back up by adding a negative space (line 3), add the penalty (line 4), and then reissue the `\glue` (line 5). In lines 6 and 7, the same method is used by `\addvspace` to add the vertical space before the heading.

Lines 8 and 9 are added by TEX when placing the actual heading text (line 10) into the galley. Note that technically the heading is considered a "paragraph", so `\parskip` is added. This is the reason why enlarging this parameter requires careful planning. The same care should be taken when adjusting other parameters (like the one added on line 7).

Side effect of `\showoutput`

The `\showoutput` command will also produce symbolic displays of overfull boxes. Tracing ends at the next closing brace or environment. Thus, to see the output for full pages, you have to ensure that the page break happens before the next group ends.

Tracing page-break decisions

If you want to trace page-breaking decisions, TeX offers symbolic information that you can turn on by setting the internal counter `\tracingpages` to a positive integer value:

```
\tracingonline=1 \tracingpages=1
```

Setting `\tracingonline` to a positive value will ensure that the tracing information will appear not only in the transcript file (default), but also on the terminal.

Processing the previous paragraph starting with "`If you want to...`" as a separate document, we get the following lines of tracing information:

```
1 %% goal height=522.0, max depth=4.0
2 % t=10.0 g=522.0 b=10000 p=150 c=100000#
3 % t=22.0 g=522.0 b=10000 p=150 c=100000#
4 % t=55.0 plus 4.0 g=522.0 b=10000 p=-51 c=100000#
5 % t=77.0 plus 8.0 g=522.0 b=10000 p=300 c=100000#
6 % t=89.0 plus 8.0 g=522.0 b=10000 p=0 c=100000#
7 % t=90.94397 plus 8.0 plus 1.0fil g=522.0 b=0 p=-10000 c=-10000#
```

The first line starting with two percent signs shows the target height for the page (i.e., `522pt` in this case), which means 43 lines at a `\baselineskip` of `12pt` with `2pt` missing since the skip to position the first base line, `\topskip`, has a value of `10pt`. If the goal height does not result in an integral number of lines, problems like underfull `\vbox`es are likely to happen.

Target size of a break

The remaining lines, starting with one percent sign, indicate a new potential page-break position that TeX has considered. You can interpret such lines as follows: `t=` shows the length of the galley so far and, if the galley contains vertical rubber spaces, their total amount of stretch and shrink. Line 4, for example, shows that in the layout of this book verbatim displays have an extra space of `10pt` plus a stretch of `4pt` (the verbatim lines are typeset in a smaller font with only `11pt` of `\baselineskip`) and the same amount is added between lines 4 and 5.

Page goal height

The `g=` specifies the goal height at this point. This value changes only if objects like floats have reduced the available space for the galley in the meantime.

Page badness

With `b=`, TeX indicates the badness of the page if a break would be taken at this point. The badness is calculated from the factor by which the available stretch or shrink in `t=` must be multiplied to reach the goal height given in `g=`. In the example the page is barely filled, so it is always `10000` (infinitely bad), except for line 7, where, due to the added `fil` stretch, the page is suddenly considered optimal (`b=0`).

Break penalty

With each breakpoint TeX associates a numerical `\penalty` as the cost to break at this point. Its value is given by `p=`. For example, it is not allowed to break directly before the verbatim display, which is why there is a large increase in `t=` between lines 3 and 4. On the other hand, a break after the display is given a bonus (p=-51). Line 5 shows that breaking after the first line of the two-line paragraph fragment following the verbatim text is considered bad (p=300), as it would result in both a club and a widow line (`\clubpenalty` and `\widowpenalty` each have a value of 150 and their values are added together).

Costs of a page break

Finally, c= describes the calculated cost to break at this breakpoint, which is derived from a formula taking the badness of the resulting page (b=) and the penalty to break here (p=) into account. TEX looks at these cost values and will eventually break at the point with minimal cost. If the line ends in #, then TEX thinks that it would be the best place to break the page after evaluating all breakpoints seen so far. In the example, all lines show this #—not surprising, given that TEX considers all but the last breakpoint to be equally bad.

If the pages would become too full if a break is taken at a particular breakpoint, then TEX indicates this fact with b=*. At this point TEX stops looking for other breakpoints and instead breaks the page at the best breakpoint seen so far.

For additional details on the output produced by these low-level display devices, consult [82, p. 112].

B.3.3 Diagnosing and solving paragraph-breaking problems

If TEX is unable to find a suitable set of points at which to break a paragraph into lines, it will, as a last resort, produce one or more lines that are "overfull". For each of them you will get a warning on the screen and in the transcript file, such as

```
Overfull \hbox (17.57108pt too wide) in paragraph at lines 3778--3793
/hlhr8t@8.80005pt/showing you a sym-bolic dis-play of the text line and the
line num-ber(s) of the paragraph|
```

showing you a symbolic display of the text line and the line number(s) of the paragraph containing it. If you look at the symbolic display, you can easily diagnose that the problem is TEX's inability[1] to hyphenate the word "paragraph". To explicitly flag such lines in your document, you can set the parameter `\overfullrule` to a positive value. For the present paragraph it was set to 5pt, producing the blob of ink clearly marking the line that is overfull. The standard document classes enable this behavior with the option `draft`. On the other hand, you may not mind lines being only slightly overfull. In that case you can change the parameter `\hfuzz` (default 0.1pt); only lines protruding by more than the value of `\hfuzz` into the margin will then be reported.

If TEX is unable to break a paragraph in a satisfying manner, the reasons are often hyphenation problems (unbreakable words, as in the above example), problems with the parameter settings for the paragraph algorithm, or simply failure of the text to fit the boundary conditions posed by the column measure or other

[1] TEX is, in fact, perfectly capable of hyphenating para-graph; for the example, we explicitly prevented it from doing so. The paragraph would have been perfect otherwise.

parameters, together with aesthetic requirements like the allowed looseness of individual lines. In the latter case the only remedy is usually a partial rewrite.

Dealing with hyphenation problems

With the relevant hyphenation patterns loaded, TEX is able to do a fairly good job for many languages [115]. However, it usually will not find all potential hyphenation points, so that sometimes one has to assist TEX in this task. To find out which hyphenation points in words like "laryngologist" are found by TEX, you can place such words or phrases in the argument of the command `\showhyphens`:

```
\showhyphens{laryngologist laryngopharyngeal}
```

Running this statement through LATEX will give you some tracing output on the terminal and in the transcript file. The hyphenation points determined by TEX are indicated by a hyphen character:

```
[] \OT1/cmr/m/n/10 laryn-gol-o-gist laryn-gopha-ryn-geal
```

If you want to add the missing hyphenation points, you can specify all hyphenation points for one word locally in the text using `\-`, for example,

```
la\-ryn\-gol\-o\-gist  la\-ryn\-go\-pha\-ryn\-ge\-al
```

Alternatively, you can use a `\hyphenation` declaration in the preamble:

```
\hyphenation{la-ryn-gol-o-gist  la-ryn-go-pha-ryn-ge-al}
```

The latter technique is particularly useful when you detect a wrong hyphenation, or often use a word for which you know that TEX misses important hyphenation points. Note that such explicit specifications tell TEX how to hyphenate words that are exactly in the form given. Thus, the plural "laryngologists" would be unaffected unless you specify its hyphenation points as well.

The `\hyphenation` declarations apply to the current language, so if a document uses several languages—for example, by using the methods provided by the babel system—then you need to switch to the right language before issuing the relevant declarations.

Tracing the paragraph algorithm

As TEX uses a global algorithm for optimizing paragraph breaking, it is not always easy to understand why a certain solution was chosen. If necessary, one can trace the paragraph-breaking decisions using the following declarations:[1]

```
\tracingparagraphs=1 \tracingonline=1
```

[1] These parameters are also turned on by a `\tracingall` command, so you may get many lines of paragraph tracing data, even if you are interested in something completely different.

For readers who really want to understand the reasons behind certain decisions, we show some example data with detailed explanations below.

Paragraph tracing will produce output that looks somewhat scary. For instance, one of the previous paragraphs generated data that starts like this:

```
1  @firstpass
2  @secondpass
3  []\T1/cmr/m/n/10 The [] dec-la-ra-tions ap-ply to the cur-rent lan-guage, so
4  @ via @@0 b=3219 p=0 d=10436441
```

Up to three passes over paragraph data

Line 2 says that TEX has immediately given up trying to typeset the paragraph without attempting hyphenation. This is due to the value of `\pretolerance` being set to `100` in the sources for the book; otherwise, TEX may have gotten further or even succeeded (in English text quite a large proportion of paragraphs can be reasonably set without hyphenating[1]). In addition to `@secondpass`, you sometimes see `@emergencypass`, which means that even with hyphenation it was impossible to find a feasible solution and another pass using `\emergencystretch` was tried.[2] Line 3 shows how far TEX had to read to find that first potential line ending that results in a badness of less than $\infty = 10000$. Line 4 gives details about this possible break. Such lines start with a single `@`; the `via` gives the previous breakpoint (in this case `@@0`, which refers to the paragraph start), the line badness (b=), the penalty to break at this point (p=), and the so-called demerits (d=) associated with taking that break (a "cost" that takes into account badness, penalty, plus context information like breaking at a hyphen or the visual compatibility with the previous line).

```
5  @@1: line 1.0 t=10436441 -> @@0
```

In line 5, TEX informs us that it would be possible to form a very loose first line ending in the breakpoint given by line 3 with a total cost (`t=`) equal to the demerits shown on line 4. This line would be formed by starting at breakpoint `@@0`. The notation `line 1.0` gives the line number being made and the suffixes `.0`, `.1`, `.2`, `.3`, respectively, stand for very loose, loose, decent, and tight interword spacing in the line. This classification is important when comparing the visual compatibility of consecutive lines.

TEX now finds more and more potential line breaks, such as after "`if`" in line 6, and after "a" in line 9. Each time TEX tells us what kind of lines can be formed that end in the given breakpoint. If `b=*` appears anywhere in the trace data, it means that TEX could not find a feasible breakpoint to form a line and had to choose an infeasible solution (i.e., one exceeding `\tolerance` for the particular line).

```
6  if
7  @ via @@0 b=1087 p=0 d=1213409
8  @@2: line 1.0 t=1213409 -> @@0
9  a
10 @ via @@0 b=334 p=0 d=128336
11 @@3: line 1.0 t=128336 -> @@0
12 doc-
13 @\discretionary via @@0 b=0 p=50 d=2600
14 @@4: line 1.2- t=2600 -> @@0
15 u-
```

[1] For the *LATEX Companion* with its many long command names this is less likely.

[2] For this to happen `\emergencystretch` needs to have a positive value. See also the discussion in Section 3.1.11.

```
16 @\discretionary via @@0 b=1 p=50 d=2621
17 @@5: line 1.2- t=2621 -> @@0
```

By hyphenating the word `doc-u-ment` it finds two more breakpoints (lines 12 and 15). This time you see a penalty of 50—the value of the parameter `\hyphenpenalty` (breaking after a hyphen)—being attached to these breaks. Line 15 is the last breakpoint that can be used to produce the first line of the paragraph. All other breakpoints would produce an overfull line. Hence, the next tracing line again shows more text; none of the potential breakpoints therein can be used as they would form a second line that exceeds `\tolerance`.

```
18 ment uses sev-eral languages---for ex-am-ple, by us-ing the meth-
19 @\discretionary via @@1 b=1194 p=50 d=1452116
20 @\discretionary via @@2 b=2875 p=50 d=8325725
21 @@6: line 2.0- t=9539134 -> @@2
```

Here the breakpoint can be used to form a second line in two different ways: by starting from breakpoint `@@1` (line 19) or by starting from breakpoint `@@2` (line 20). If we compare just these two solutions to form the second line of the paragraph, then the first would be superior: it has a badness of `1194`, whereas the second solution has a badness of `2875`, which results in a factor of 5 in "costs" (`d=`). Nevertheless, TeX considers the second break a better solution, because a first line ending in `@@1` is so much inferior to a line ending in `@@2` that the total cost for breaking is less if the second alternative is used. TeX therefore records in line 21 that the best way to reach the breakpoint denoted by line 18 is by coming via `@@2` and results in a total cost of `t=9539134`. For the rest of the processing, TeX will not need to know that there were several ways to reach `@@6`; it just needs to record the best way to reach it.

More precisely, TeX needs to record the best way to reach a breakpoint for any of the four types of lines (very loose, loose, decent, tight), since the algorithm attaches different demerits to a solution if adjacent lines are visually incompatible (e.g., a loose line following a tight one). Thus, later in the tracing (lines 22–40 are not shown), we get the following output:

```
41 by
42 @ via @@3 b=19 p=0 d=10841
43 @ via @@4 b=9 p=0 d=361
44 @ via @@5 b=42 p=0 d=2704
45 @@10: line 2.1 t=5325 -> @@5
46 @@11: line 2.2 t=2961 -> @@4
```

This output indicates that there are three ways to form a line ending in "`by`": by starting from `@@3`, `@@4`, or `@@5`. A line with a badness of 12 or less is considered decent (suffix `.2`); a line stretching, but with a badness not higher than 100, is considered loose (suffix `.1`). So here TeX records two feasible breakpoints for further consideration—one going through `@@5` and one going through `@@4`.

Which path through the breakpoints is finally selected will be decided only when the very end of the paragraph is reached. Thus, any modification anywhere in the paragraph, however minor, might make TeX decide that a different set of breakpoints will form the best solution to the current line-breaking problem, because it will produce the lowest total cost. Due to the complexity of the algorithm, minor modifications sometimes have surprising results. For example, the deletion of a word may make the paragraph a line longer. This may happen because TeX decides that using uniformly loose lines, or avoiding hyphenation of a word, is preferable to some other way to break the paragraph. Further details, describing all parameters that influence the line-breaking decisions, can be found in [82, p.98]. If necessary, you can force breakpoints in certain places with `\linebreak`, or prevent them with `\nolinebreak` or by using `~` in place of a space. Clearly, choices in the early parts of a paragraph are rather limited and you may have to rewrite a sentence to avoid a bad break. But later in a paragraph nearly every potential break will become feasible, being reachable without exceeding the specified `\tolerance`.

Shortening or lengthening a paragraph

Another low-level tool that can be used is the internal counter `\looseness`. If you set it to a nonzero integer n, TEX will try to make the next paragraph n lines longer (n positive) or shorter (n negative), while maintaining all other boundary conditions (e.g., the allowed `\tolerance`). In fact, the last paragraph of the previous section was artificially lengthened by one line by starting it in the following way:

```
\looseness=1
Which path through the breakpoints is finally selected
```

Setting the value of `\looseness` is not guaranteed to have any effect. Shortening a paragraph is more difficult for TEX than lengthening it, since interword spaces have a limited shrinkability that is small in comparison to their normal stretchability. The best results are obtained with long paragraphs having a short last line. Consequently, extending a paragraph works best on long paragraphs with a last line that is already nearly full, though you may have to put the last words of the paragraph together in an `\mbox` to ensure that more than one word is placed into the last line.

B.3.4 Other low-level tracing tools

TEX offers a number of other internal integer parameters and commands that can sometimes help in determining the source of a problem. They are listed here with a short explanation of their use.

On-line tracing

We already encountered `\tracingonline`. If it is set to a positive value all tracing information is shown on the terminal; otherwise, most of it is written only to the transcript file. This parameter is automatically turned on by `\tracingall`.

With `\tracingoutput`, tracing of page contents is turned on. What is shown depends on two additional parameters: `\showboxdepth` (up to which level nested boxes are displayed) and `\showboxbreadth` (the amount of material shown for each level). Anything exceeding these values is abbreviated using `etc.` or `[]` (indicating a nested box) in the symbolic display. The LATEX command `\showoutput` sets these parameters to their maximum values and `\tracingoutput` to 1, so that you get the most detailed information possible. The `\showoutput` command is automatically called by `\tracingall`.

The contents of boxes

To see the contents of a box produced with `\sbox` or `\savebox`, you can use the TEX command `\showbox`:

```
\newsavebox\test \sbox\test{A test} {\tracingonline=1 \showbox\test }
```

However, the result is fairly useless if you do not adjust both `\showboxdepth` and `\showboxbreadth` at the same time. Hence, a better strategy is to use LATEX's `\showoutput`:

```
{\showoutput \showbox\test }
```

Notice the use of braces to limit the scope of `\showoutput`. Without the braces you would see all of the following page boxes, which might not be of much interest. The same type of symbolic display as discussed in Section B.3.2 will be displayed on the terminal:

```
> \box26=
\hbox(6.83331+0.0)x27.00003
.\OT1/cmr/m/n/10 A
.\glue 3.33333 plus 1.66498 minus 1.11221
.\OT1/cmr/m/n/10 t
.\OT1/cmr/m/n/10 e
.\OT1/cmr/m/n/10 s
.\OT1/cmr/m/n/10 t
```

If you add `\scrollmode` or `\batchmode` before the `\showbox` command, LATEX will not stop at this point. You can then study the trace in the transcript.

Local restores

To see what values and definitions TEX restores when a group ends, you can set `\tracingrestores` to a positive value. It is automatically turned on by `\tracingall`.

TEX's stack of lists

With `\showlists` you can direct TEX to display the stack of lists (vertical, horizontal) that it is currently working on. For instance, putting `\showlists` into the footnote[1] of the present paragraph, we obtain the following output in the transcript file:

```
### horizontal mode entered at line 3066 []
spacefactor 1000
### internal vertical mode entered at line 3066
prevdepth ignored
### horizontal mode entered at line 3060 []
spacefactor 1000
### vertical mode entered at line 0
### current page: []
total height 514.70349 plus 26.0 minus 2.0
 goal height 522.0
prevdepth 1.70349
```

Here the text of the footnote started at line 3066 and the `\spacefactor` was set to 1000 at its beginning. The footnote itself was started on that same line, contribut-

[1] A footnote starts a new vertical list and, inside it, a new horizontal list for the footnote text.

ing the "internal vertical mode", and TEX correctly disregarded the outer value of \prevdepth. The footnote was part of a paragraph that started on source line 3060, which in turn was embedded in a vertical list that started on line 0, indicating that it is the main vertical galley. Finally, the output shows some information about the current page list that is being built, including its current height, its target height, and the value of \prevdepth (i.e., the depth of the last line on the page at the moment).

Because of the default settings for \showboxbreadth and \showboxdepth, the contents of all lists are abbreviated to []. To get more detail adjust them as necessary or use \showoutput\showlists to get the full details.

Tracing the processing

Not very useful on its own, but helpful together with other tracing options, is \tracingcommands, which shows all primitives used by TEX during processing. A related internal integer command is \tracingmacros, which shows all macro expansions carried out by TEX. If set to 2, it will also display the expansion of conditionals. Both parameters are automatically turned on by \tracingall.

Tracing lost characters

When everything is set up correctly, it is unlikely that TEX will ever access a font position in the current font that is not associated with a glyph. However, some commands, such as \symbol, can explicitly request any font slot, so it is not impossible. Unfortunately, TEX does not consider this event to be an error (which it should). It merely traces such missing characters by writing unsuspicious transcript entries, and it takes that step only if \tracinglostchars is set to a positive value. LATEX tries to be helpful by initializing this internal integer to 1.

Stepping through a document

Finally, you can direct TEX to step through your files line by line. When setting \pausing to 1, each source line is first displayed (suffixed with =>). TEX then waits for instructions regarding what to do with it. Pressing ⟨*Enter*⟩ instructs TEX to use the line unchanged; anything else means that TEX should use the characters entered by the user instead of the current line. TEX then executes and typesets whatever it was passed, displays the next line, and stops again. To continue normal processing you can reply with \pausing=0, but remember that this is used in place of the current source line, so you may have to repeat the material from the current source line as well.

B.3.5 trace—Selectively tracing command execution

The LATEX command \tracingall (inherited from plain TEX) is available to turn on full tracing. There are, however, some problems with this command:

1. There is no corresponding command to turn off tracing. As a consequence, you have to delimit the scope, which is not always convenient or even possible.
2. Some parts of LATEX produce enormous amounts of tracing data that is of little or no interest for the problem at hand.

For example, if LATEX has to load a new font, it enters some internal routines of NFSS that scan font definition tables and perform other activities. And 99.9% of the time you are not at all interested in that part of the processing, but just in the two lines before and the five lines after it. Nevertheless, you have to scan through a few hundred lines of output and try to locate the lines you need (if you can find them).

Another example is a statement such as `\setlength\linewidth{1cm}`. With standard LaTeX this gives 5 lines of tracing output. With the calc package loaded, however, it will result in about 60 lines of tracing data—probably not what you expected and not really helpful unless you try to debug the calc parsing routines (which ideally should not need debugging).

To solve the first problem, the trace package [122] by Frank Mittelbach defines a pair of commands, `\traceon` and `\traceoff`. If LaTeX is used on top of a TeX engine, then `\traceon` is essentially another name for `\tracingall`: it turns on the same tracing switches (albeit in a different order to avoid tracing itself). If LaTeX is run on top of the eTeX engine, then the tracing of assignments and groups is also turned on.[1]

More tracing info available with eTeX

Another difference between `\traceon` and `\tracingall` is that the latter will always display the tracing information on the terminal, whereas `\traceon` can be directed to write only to the transcript file if you specify the option `logonly`. This is useful when writing to the terminal is very slow (e.g., if running in a shell buffer inside emacs).

To solve the second problem, the trace package has a number of internal commands for temporarily disabling tracing. It redefines the most verbose internal LaTeX functions so that tracing is turned off while they are executing. For example, the function to load new fonts is handled in this way. If a document starts with the two formulas

```
$a \neq b$   \small  $A = \mathcal{A}$
```

then LaTeX will load 22 new fonts[2] at this point. Using standard `\tracingall` on that line will result in roughly 7500 lines of terminal output. On the other hand, if `\traceon` is used, only 350 lines will be produced (mainly from tracing `\small`).

The commands for which tracing is turned off are few and are unlikely to relate to the problem at hand. However, if you need full tracing, you can either use `\tracingall` or specify the `full` option. In the latter case, `\traceon` traces everything, but you can still direct its output exclusively to the transcript file.

[1] The corresponding eTeX switches are `\tracingassigns` and `\tracinggroups`; see [27].

[2] You can verify this with the `loading` option of the tracefnt package.

Appendix C

LaTeX Software and User Group Information

The files and packages that are described in this book are available in most TeX distributions, such as the TeX Live CD-ROM (provided with this book), or on the CTAN CD-ROMs of DANTE. The newest versions can also be directly downloaded from the web. The aim of this appendix is to provide you with the necessary information to obtain current releases of these CD-ROMs and to give hints on how to locate and get the files you need directly from the Internet.

C.1 Getting help

While we certainly hope that your questions have been answered in this book, we know that this cannot be the case for all questions. For questions related to specific packages discussed in the book, it can be helpful to read the original documentation provided with the package. Appendix C.4 suggests ways to find that documentation on your system.

Very valuable resources are the existing FAQ documents. The most important ones are the UK-TUG FAQ by Robin Fairbairns available at `http://www.tex.ac.uk/faq` and the DANTE FAQ by Bernd Raichle et al. available at `http://www.dante.de/faq/de-tex-faq` (in German). Robin's FAQ is also available in HTML format on the CD-ROM in the directory `/texmf/doc/html/faq/index.html`. However, as both documents are constantly being developed further, it is best to access the on-line versions if possible.

If precomposed answers are not enough, several news groups are devoted to general TeX and LaTeX questions: `news://comp.text.tex` is perhaps the most important one, with usually more than 100 messages posted each day. Many of the authors mentioned in this book are regular contributors on the news groups and help with answering questions and requests. Thus, there is a vast amount of helpful material on the web that can be conveniently searched using any search engine that indexes news entries.

If you post to any of these news groups, please adhere to basic netiquette. The community is friendly but sometimes direct and expects you to have done some research of your own first (e.g., read the FAQ first and searched the archived news) and not ask questions that have been answered several hundred times before. You should perhaps read Eric Raymond's "*How To Ask Questions The Smart Way*", available at `http://www.catb.org/~esr/faqs/smart-questions.html`, as a starter. Also, if applicable, provide a minimal *and* usable example of your problem that allows others to easily reproduce the symptoms you experience—this will save others time and might get you a faster reply.

C.2 How to get those TeX files?

A useful entry point to the TeX world is the TeX Users Group home page (`http://www.tug.org`; see Figure C.1). From there you can reach most information sources about TeX and friends available worldwide.

In particular, from the TeX Users Group home page you can go to one of the CTAN (Comprehensive TeX Archive Network) nodes. CTAN is a collaborative effort initiated in 1992 by the TeX Users Group Technical Working Group on TeX Archive Guidelines originally coordinated by George Greenwade, building on earlier work of Peter Abbott (see [61] for the historical background), and currently maintained by Jim Hefferon, Robin Fairbairns, Rainer Schöpf, and Reinhard Zierke (spring 2004). Its main aim is to provide easy access to up-to-date copies of all versions of TeX, LaTeX, METAFONT, and ancillary programs and their associated files.

Presently, there are three backbone machines that act as FTP servers: in the United Kingdom (`cam.ctan.org` or `ftp.tex.ac.uk`), in Germany (`dante.ctan.org` or `ftp.dante.de`), and in the United States (`tug.ctan.org` or `ctan.tug.org`). Moreover, these sites are mirrored worldwide.

The material on CTAN is regularly (currently on a yearly basis) made available on CD-ROMs. One is the TeX Live distribution ([157]; see also `www.tug.org/texlive`), which provides a "runnable" version of TeX for various platforms. TeX Live CD-ROMs have been developed since 1996 through a collaboration between the TeX Users Group (TUG; United States) and the TeX user groups of the Czech Republic, France, Germany, India, Netherlands, Poland, Slovakia, and the United Kingdom. These user groups distribute the TeX Live CD-ROMs to their members, so you should contact them directly (their addresses are given in Section C.5).

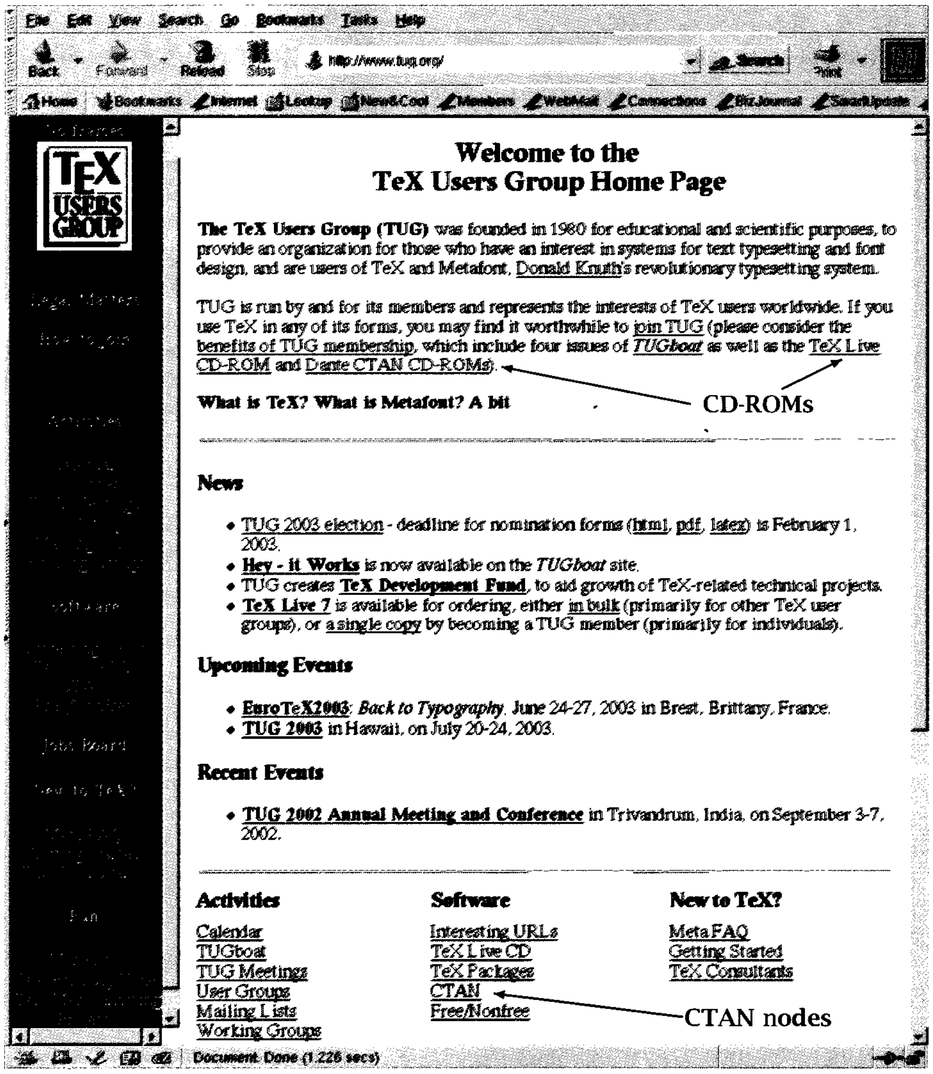


Figure C.1: The TEX Users Group web home page

Another distribution is prepared by the German-speaking TEX user group DANTE (see Section C.5) and contains on several (three in October 2002) CD-ROMs an image of the complete CTAN file tree (almost 2 GB of data). Much like the TEXlive CD-ROMs, these DANTE CTAN CD-ROMs are distributed by most user groups to their members. Thus, the same procedure as for TEXlive should be used if you are interested in getting a copy of the set.

C.3 Using CTAN

In the previous section we described the TeX Live and DANTE CTAN CD-ROM sets. Obtaining the latest version of these CD-ROMs is an optimal way for getting access to recent versions of LaTeX software.

Nevertheless, for those readers with an Internet connection, it makes sense to query one of the CTAN nodes every now and then to see whether one of the LaTeX components you need has been updated. If you find updates, you can download the latest version of the given package directly from a CTAN archive or one of its mirror sites.

Although network connections get faster all the time, it is often wise to connect to a site that is not too distant geographically from your location. The most convenient way to make an Internet connection is via a web browser, especially since user-friendly interfaces to the CTAN archives have been developed.

C.3.1 Finding files on the archive

The easiest way to find a file on CTAN is using a web interface (the TeX Users Group home page proposes a list of search engines at `http://www.tug.org/ctan.html`). For instance, we use Peter Flynn's server in Ireland, which lets you choose the CTAN node you want to connect to (see Figure C.2).

We connect to the DANTE CTAN Internet node at `http://ftp.dante.de` (upper oval in Figure C.2) and specify the search string "`graphicx`" (second oval in Figure C.2). The search engine returns the list of all files in the CTAN archive matching the given search criterion (for greater clarity, the result of our search is shown inside the rectangle in Figure C.2). By clicking on one of these links you can view (or download) the file in question (in this case we decided to view the file `graphicx.dtx`, whose beginning is shown in the browser window at the bottom of Figure C.2).

C.3.2 Using the TeX file catalogue

A catalogue of TeX- and LaTeX-related packages maintained by Graham Williams can be consulted at `http://datamining.csiro.au/tex/catalogue.html`. This interface is especially attractive when you are "surfing" the archive and want to know what a certain package does. Moreover, the interface lets you choose the site from which to download the software, so that you can optimize your connection. When an extended description of the package is available on the Internet, its URL is presented together with the relevant entry.

Several instances of this catalogue, which is regularly updated by its author, are available on the Internet. You can choose the instance closest to where you reside by clicking on one of the flags displayed at the beginning of the catalogue page (see Figure C.3).

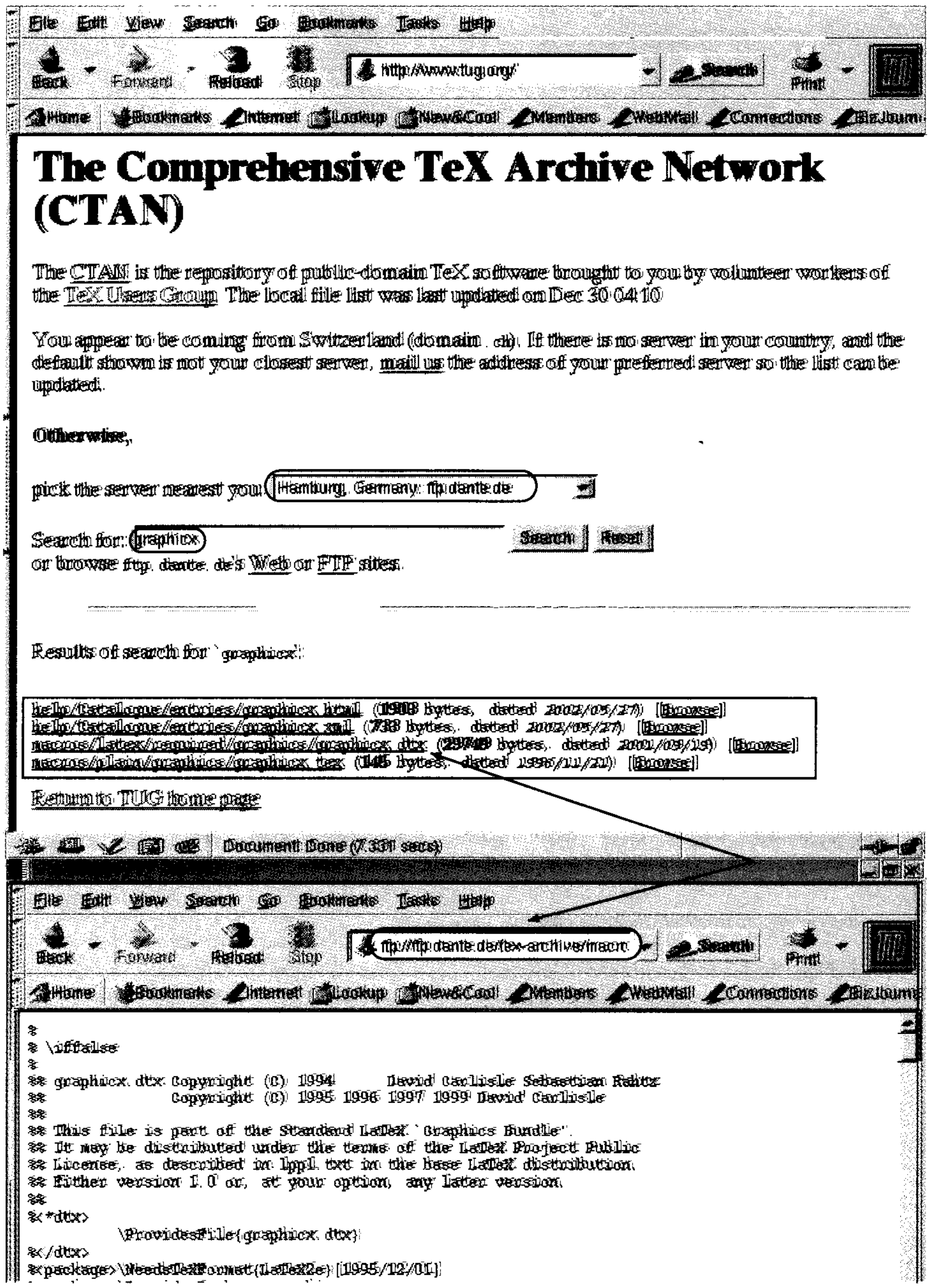


Figure C.2: Using the CTAN web interface

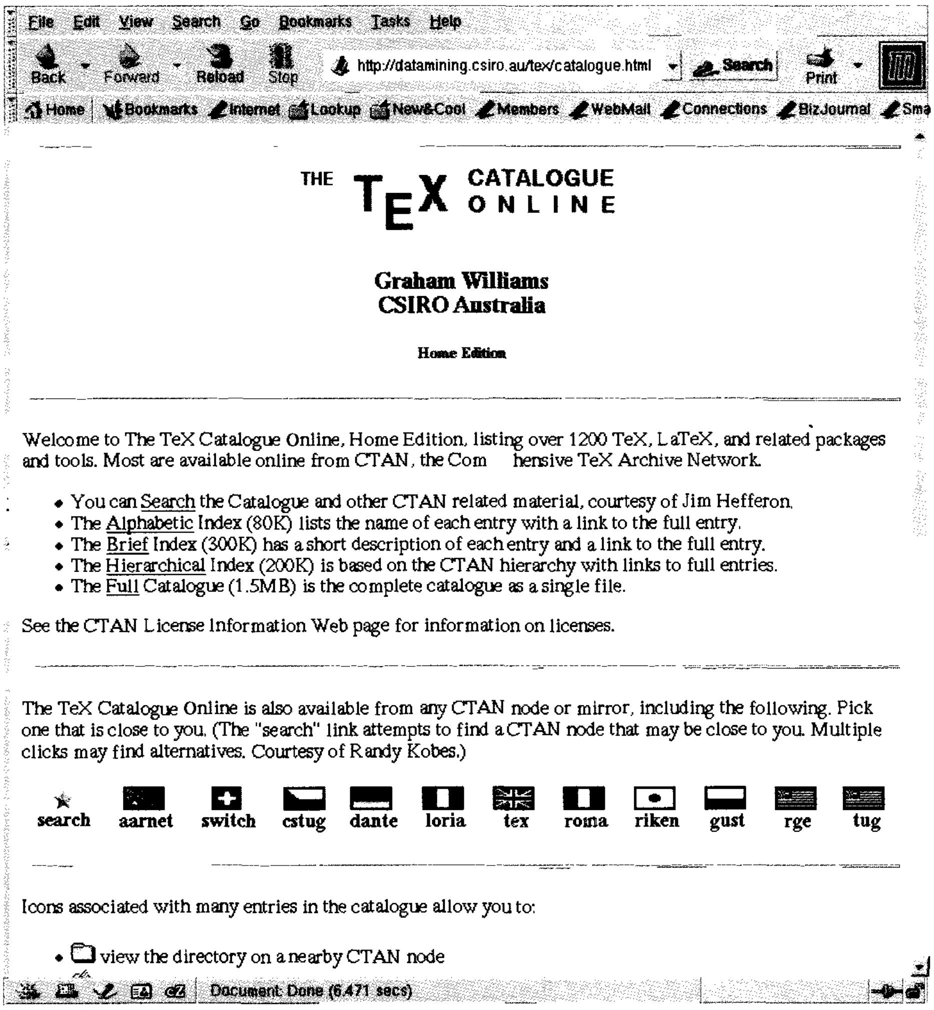


Figure C.3: Graham Williams' TEX catalogue on the web

C.3.3 Getting multiple files

Web interfaces are very handy when you want to check the characteristics of one or more specific files (e.g., date of last modification, size, purpose), but they are not particularly convenient for transferring complete packages consisting of many (sometimes hundreds of) files. In this case it is more appropriate to connect directly via FTP to one of the CTAN nodes or their mirrors. Below is a typical interactive session (commands input by the user are underlined):

```
> ftp ftp.dante.de
Connected to ftp.dante.de (134.100.9.51).
220 nova.dante.de FTP server (Version wu-2.6.2(1) Sat Dec 1 07:52:37 CET 2001).
Name (ftp.dante.de:goossens): ftp
331 Guest login ok, send your complete e-mail address as password.
Password: uuu.vvv@xxx.zz (use your email address here!)
230-Welcome, archive user!  This is an FTP server for the DANTE Archive.
   ...
ftp> quote site index graphicx
200-index graphicx
200-NOTE. This index shows at most 20 lines. for a full list of files,
200-retrieve /tex-archive/FILES.byname
200-2002/05/27 |       1903 | help/Catalogue/entries/graphicx.html
200-2002/05/27 |        733 | help/Catalogue/entries/graphicx.xml
200-2001/09/19 |      29749 | macros/latex/required/graphics/graphicx.dtx
200-1996/11/21 |        145 | macros/plain/graphics/graphicx.tex
200  (end of 'index graphicx')
ftp> bin
200 Type set to I.
ftp> cd ctan:
250-Machine specific implementations      --> systems
250-Original Knuthian sources:            --> systems/knuth.
250-LaTeX styles, plain macros, MusicTeX: --> macros.
250-LaTeX2e:                              --> macros/latex.
   ...
ftp> cd macros/latex/required
250-Please read the file README
250-  it was last modified on Wed Mar 24 01:00:00 1999 - 1376 days ago
250 CWD command successful.
ftp> get graphics.zip
local: graphics.zip remote: graphics.zip
227 Entering Passive Mode (134,100,9,51,245,92)
150 Opening BINARY mode data connection for /bin/ZIP.
226 Transfer complete.
127985 bytes received in 0.76 secs (1.6e+02 Kbytes/sec)
ftp> get graphics.tar.gz
local: graphics.tar.gz remote: graphics.tar.gz
227 Entering Passive Mode (134,100,9,51,63,61)
150 Opening BINARY mode data connection for /bin/TARZ.
226 Transfer complete.
118125 bytes received in 0.569 secs (2e+02 Kbytes/sec)
ftp> quit
221-You have transferred 246110 bytes in 2 files.
221-Total traffic for this session was 250149 bytes in 2 transfers.
221-Thank you for using the FTP service on nova.dante.de.
221 Goodbye.
```

After connecting to the site (`ftp.dante.de` above), you should specify `ftp` as login name and type *your* e-mail address as the password. Then you can send a query with the command "`quote site index` ⟨*term*⟩", where ⟨*term*⟩ is the query string. We submitted the same query for the term "`graphicx`" as in Figure C.2 on page 951; the result is, of course, consistent with the contents of the rectangle in the middle part of that figure. We now decide to transfer the graphics package, so we first position ourself at the root of the CTAN archive tree by issuing the command "cd ctan:" (note the colon at the end, which is necessary!). From our query we know where in the tree the files we want are located, so we change to the directory just one level above (cd macros/latex/required). Then we transfer the directory twice: once as a .zip file and once as a compressed .tar archive (just to show how to specify the commands). The command "quit" ends the FTP session.

C.4 Finding the documentation on your TeX system

When you want to use a LaTeX package, it would be nice if you could study the documentation without having to remember where the relevant files are located on your TeX system. Two ways exist to help you in your search: texdoc and its derivative texdoctk.

C.4.1 texdoc—Command-line interface for a search by name

Thomas Esser developed the program texdoc, which is part of the TeXlive distribution. If you know the name of the file describing a package, you can find the relevant documentation files as follows:

```
texdoc -l pspicture
/TeXlive/tl7/texmf/doc/latex/carlisle/pspicture.dvi
/TeXlive/tl7/texmf/doc/html/catalogue/entries/pspicture.html
```

The -l option tells texdoc to list only the path to the files that fulfill the selection criterion (in this case, files called pspicture regardless of their extension). If you do not specify the -l option, texdoc will show you the contents of the documentation file (in this case, pspicture.dvi) with the help of the relevant display program (for instance, xdvi or Windvi).

If you do not know the precise name of the file, you can specify the -s option and provide a wildcard-like specification as a search pattern. For instance:

```
texdoc -s *picture*
/TeXlive/tl7/texmf/doc/generic/mfpic/examples/lapictures.tex
/TeXlive/tl7/texmf/doc/generic/mfpic/examples/pictures.tex
/TeXlive/tl7/texmf/doc/latex/carlisle/pspicture.dvi
/TeXlive/tl7/texmf/doc/html/catalogue/entries/pspicture.html
/TeXlive/tl7/texmf/doc/html/catalogue/entries/pspicture.xml
```

Here we have picked up files that have the string picture in their name—among them the "pspicture" files we found previously.

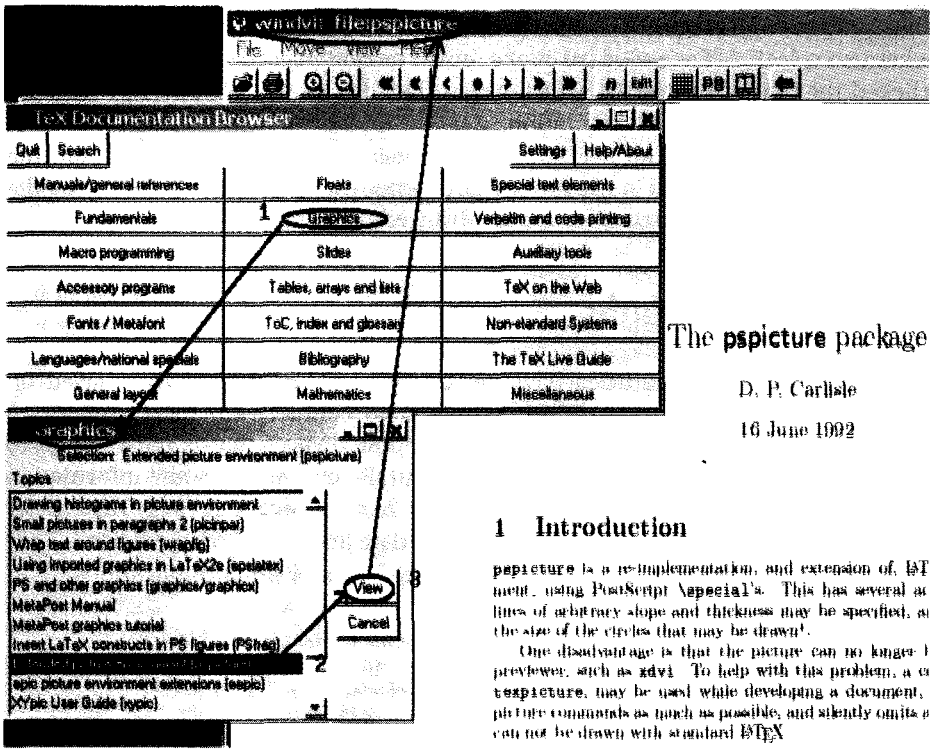


Figure C.4: Finding documentation with the texdoctk program

The texdoc utility is quite useful, but it has a drawback: you must know the name of the file describing the package that you want to use. This is not always just the name of the package itself (as with pspicture in the above examples).

C.4.2 texdoctk—Panel interface for a search by subject

Thomas Ruedas took a somewhat different approach to provide easy access to the documentation for files present on your TEX system. His texdoctk program uses a graphics user interface based on perl and Tk. The program uses a database that groups documentation files present in the Thomas Esser's tetex distribution (TEX Live is based on tetex) into 17 categories, and offers an eighteenth "user's" category to allow users to add their (local) documentation entries into the database, if needed. As with texdoc, the display or print programs present on the system will be used for viewing (e.g., xdvi, dvips).

Figure C.4 shows how we used the texdoctk system to display the documentation for the pspicture package. In this case we did not have to know the name of the package. In fact, we navigated from the main panel, where we chose the "`Graphics`" category (1), which opened the "`Graphics`" menu (lower left), where we selected "`Extended picture environment (pspicture)`" (2). We then clicked the "`View`" button (3), which called the `.dvi` viewer Windvi (4), which displayed the text of the documentation.

On the figure one can see all available documentation categories (note the "`Miscellaneous`" button in the lower-right corner for special cases) as well as the "`Search`" and "`Help`" buttons for more advanced use.

C.5 TeX user groups

TeX users in several countries have set up TeX user groups, mostly based on language affinities. If you need help, you should contact your local user group first, since they might be able to come up with an answer that is most suited to your language-dependent working environment. Below we give some information about groups that have a formal existence (see `http://www.tug.org/lugs.html` or `http://www.servalys.nl/lug/` for up-to-date and more complete lists). They can help you obtain TeX-related material on CD-ROMs or other publications.

cn: **China PR**
name: Chinese TeX Users Group
language: Chinese
web site: `www.rons.net.cn`
name: Hong Feng
address: RON's Datacom Co., Ltd.
79, DongWu Ave.,
Wuhan, Hubei Province
430040 China P.R.
e-mail: `info@mail.rons.net.cn`
phone: +862783222108
fax: +862783222108

cz: **Czech Republic**
language: Czech
name: CsTUG
contact: Petr Sojka
address: CsTUG, c/o FI MU
Botanická 68a
CZ-602 00 Brno
Czech Republic
e-mail: `cstug@cstug.cz`
web site: `www.cstug.cz`
phone: +420541212352

de: **Germany**
name: DANTE e.V.
language: German
contact: Volker Schaa
address: Postfach 101840
D-69008 Heidelberg
Germany
e-mail: `dante@dante.de`
web site: `www.dante.de`
phone: +49622129766
fax: +496221167906

dk: **Denmark**
name: DK-TUG
language: Danish
contact: Kaja Christiansen
address: Department of Computer Science
Ny Munkegade, Bldg. 540
DK-8000 Århus C
Denmark
e-mail: `board@tug.dk`
web site: `www.tug.dk`
phone: +4589423220

ee: **Estonia**
name: Estonian User Group
address: Astrophysical Observatory,
Toravere
Enn Saar, Tartu
EE 2444 Estonia
e-mail: `saar@aai.ee`

es: **Spain (CervanTeX)**
name: CervanTeX
language: Spanish
e-mail: `secretario@cervantex.org`
web site: `www.cervantex.org`

esc: **Spain (Catalan)**
name: Catalan TeX Users Group
language: Catalan

contact: Gabriel Valiente
address: Technical University of Catalonia
Jordi Girona Salgado, 1-3
E-08034 Barcelona
Spain
e-mail: `valiente@lsi.upc.es`
web site: `www-lsi.upc.es/~valiente/tug-catalan.html`

fr: France
name: GUTenberg
language: French
address: c/o Irisa
Campus Universitaire de Beaulieu
F-35042 Rennes cedex
France
e-mail: `gut@irisa.fr`
web site: `www.gutenberg.eu.org`
phone: +33681665102
fax: +33492579667

fra: France (Astex)
short name: AsTEX
language: French
address: Association AsTEX
BP 6532
45066 Orleans cedex 2
France
e-mail: `astex-admin@univ-orleans.fr`
web site: `www.univ-orleans.fr/EXT/ASTEX/astex/doc/en/web/html/astex000.htm`
phone: +33238640994

gr: Greece
name: Greek TeX Friends Group
language: Greek
contact: Apostolos Syropoulos
address: 366, 28th October Str.
GR-671 00 Xanthi
Greece
e-mail: `eft@ocean1.ee.duth.gr`
web site: `obelix.ee.duth.gr/eft/`
phone: +3054128704

hu: Hungary
name: MaTeX
language: Hungarian
address: Institute of Mathematics and Informatics
University of Debrecen
H-4010 Debrecen, P.O. Box 12
Hungary
e-mail: `matex@math.klte.hu`
web site: `www.math.klte.hu/~matex/`

in: India
name: TUGIndia
contact: K.S.S. Nambooripad
address: Kripa, TC 24/548, Sastha Gardens
Thycaud, Trivandrum 695014
India
e-mail: `tugindia@river-valley.com`
web site: `www.river-valley.com/tug/`
phone: +91471324341
fax: +91471333186

kr: Korea
name: KTUG
language: Korean
contact: Kim Kangsu
e-mail: `info@mail.ktug.or.kr`
web site: `www.ktug.or.kr`

lt: Lithuania
name: Lietuvos TeX'o Vartotojų Grupė
contact: Vytas Statulevicius
address: Akademijos 4
LT-2600 Vilnius
Lithuania
e-mail: `vytass@ktl.mii.lt`
phone: +3702359609
fax: +3702359804

mx: Mexico
name: TeX México
address: Rayon No. 523, Centro 58000
Morelia, Michoacan
Mexico
e-mail: `tex@ciencia.dcc.umich.mx`
web site: `ciencia.dcc.umich.mx./tex/`
phone: +52143128724
fax: +52143173945

nl: Netherlands, Belgium (Flemish part)
name: NTG
language: Dutch
contact: Hans Hagen
address: Pragma
Ridderstraat 27
8061 GH Hasselt
The Netherlands
e-mail: `info@ntg.nl`
web site: `www.ntg.nl`
phone: +31384775369
fax: +31384775374

no: Nordic countries
name: NTUG
language: Scandinavian languages
discussion: `nordictex@ifi.uio.no`
contact: Dag Langmyhr
address: University of Oslo
PO Box 1080 Blindern
N-0316 Oslo
Norway
e-mail: `dag@ifi.uio.no`

web site: www.ifi.uio.no/~dag/ntug/
phone: +4722852450
fax: +4722852401

ph: Philippines
name: TUG-Philippines
contact: Felix P. Muga II
address: Ateneo de Manila University
Loyola Heights
Quezon City
Philippines
e-mail: fpmuga@admu.edu.ph
phone: +6324266001 ext 2515
fax: +6324266008

pl: Poland
name: GUST
language: Polish
address: UCI UMK
Gagarina 7
87-100 Toruń
Poland
e-mail: sekretariat@gust.org.pl
web site: www.GUST.org.pl

pt: Portugal
name: GUTpt
language: Portuguese
contact: Pedro Quaresma de Almeida
address: Coimbra University
Dep. Matemática, Largo D.Dinis
Apartado 3008, 3001-454
COIMBRA
Portugal
e-mail: GUTpt@hilbert.mat.uc.pt
web site: http://hilbert.mat.uc.pt/~GUTpt/
phone: +351239791181

ru: Russia
name: CyrTUG
e-mail: cyrtug@mir.msk.su
web site: www.cemi.rssi.ru/cyrtug/
discussion: CyrTeX-en@vsu.ru
subscription: CyrTeX-en-on@vsu.ru

si: Slovenia
name: TeXCeH
contact: Vladimir Batagelj
address: Jadranska 19
SI-61111 Ljubljana
Slovenia
e-mail: Tex.Ceh@fmf.uni-lj.si
web site: vlado.fmf.uni-lj.si/texceh/texceh.htm

uk: United Kingdom
name: UKTUG
language: British English
e-mail: uktug-enquiries@tex.ac.uk
web site: uk.tug.org
contact: Dr R.W.D. Nickalls
address: Department of Anæsthesia
Nottingham City Hospital NHS Trust
Hucknall Road
Nottingham, NG5-1PB (UK)
e-mail: enxtw1@nottingham.ac.uk
phone: +441159691169 (ext. 45637)
fax: +441159627713

us: TeX User Group (international)
name: TUG
address: P.O. Box 2311
Portland, OR 97208-2311
U.S.A
e-mail: office@tug.org
web site: www.tug.org
phone: +15032239994
fax: +15032233960

vn: Vietnam
name: ViêtTUG
contact: Nguyên-Đai Quý
address: LTAS-University of Liège
Rue des Chevreuils, 1
Bât B52, Local 522B
B4000, Liège
Belgium
e-mail: viettug@eGroups.com
phone: +3243669098
fax: +3243669311

Appendix D

TLC2 TEX CD

The CD-ROM at the back of this book will enable you to set up a LaTeX system that is as close as possible to the descriptions in this book. This appendix explains how we created this CD and gets you started on how to use it.

Origins—The TEX Live system

TEX Live is an "open source" distribution of TEX and LaTeX that is sponsored by an international consortium of TEX user groups. The TLC2 TEX CD-ROM is based closely on this distribution and we therefore wish to thank all the individuals involved in the production and maintenance of TEX Live over the years.

The 2003 release of the TEX Live distribution was distributed as three disks: a DVD containing the full distribution and a copy of the CTAN archives, a CD containing (in compressed form) a full TEX Live distribution, and a "demo" disk containing a TEX distribution that may be either installed on a hard disk, or used directly from the CD.

To fit onto one CD, some packages had to be omitted from the "demo" CD, and only the major machine architectures are supported: Linux, Windows and MacOSX.

The TLC2 TEX CD-ROM is a version of the TEX Live "demo" CD. All the binary programs are unchanged, several packages described in this book have been updated or added, and the LaTeX format itself is the 2003/12/01 release. In order to keep within the size constraint, some packages had to be removed. A full list of changed packages is contained in the file `readme-tlc2.html`, which can be found in the top level directory on the CD.

Installing LaTeX from the CD-ROM

Installation and use of this CD-ROM follows exactly the procedures outlined for the TeX Live demo distribution from the original TeX Live documentation. An overview of these procedures is in the file `readme.html`, which has links to more extensive documentation files on the CD. (Much of the TeX Live documentation is available in several languages.)

In brief, the install script install-tl.sh in the top level directory should be run by you on Linux or MacOSX. Under Windows the Install program should automatically start (or double click on `autorun.exe`). This process will lead you through some configuration options and then install a LaTeX system on your hard disk. Depending on the options chosen, some lesser used packages may not be installed initially; they may be added to your local installation later, as described in the TeX Live documentation.

If you are already using LaTeX then you may not want to install the whole system but simply use the CD to update your base LaTeX and your chosen packages to more recent versions.

Running LaTeX directly from the CD-ROM

As an alternative to installing the whole system on your local disk, you can opt to run all software directly from the CD-ROM. However, some local disk space will still be required so that TeX can write output files and, if necessary, extra fonts can be generated.

Under Windows this option is taken by choosing the `Explore CD-Rom/Run TeX off CD-Rom` menu option from the TeX Live welcome program. On the other systems you should run install-tl.sh as above, but choose the option to run directly from the media.

In addition to giving you a running TeX system, this installation will also set up xemacs as an environment for preparing your documents. This provides an extensive set of menu options to help in the editing of LaTeX documents, and in the use of LaTeX and associated programs such as BibTeX.

The LaTeX Companion example documents

Files for all the examples displayed in the book are on the CD-ROM in the directory `Books/tlc2/examples`. The file name is in each case the example number, with extension `.ltx` or `.ltx2` (for two-page examples), as in `1-3-1.ltx` and `2-4-4.ltx2`.

Most of these examples use the class file ttctexa.`cls` which is in the same directory as the examples. This class is a small extension of the article class: it defines some extra commands to control the display of preamble commands in this book.

If the TEX system is used directly from the CD then all those packages required for the examples will be available, with the exception of some packages which relate to commercial fonts that cannot be distributed on this CD-ROM.

If the distribution is installed on a hard disk then not all the packages are installed by default. Extra individual packages can be installed using either install-pkg.sh under Linux and MacOSX, or the TEX Live/maintenance option from the Start menu under Windows.

Licenses

The file LICENSE.TL in the top level directory describes the license and copying conditions for TEX Live itself; these also apply to the modified distribution on the TLC2 TEX CD-ROM. All the software contained on this CD-ROM is (to the best of our knowledge) freely distributable, although different licenses are used on the different components, as detailed in the documentation of each package.

Many of the LATEX packages, and all of the example files for this book, are distributed under the LATEX Project Public License, the text of which is on the CD in the file texmf/doc/latex/base/lppl.txt.

The LPPL allows arbitrary use, including copying and modification, so long as you do not distribute modified copies with the same name as the original files.

Bibliography

[1] Adobe Systems Incorporated. Adobe Type 1 Font Format. Addison-Wesley, Reading, MA, USA, 1990. ISBN 0-201-57044-0.
The "black book" contains the specifications for Adobe's Type 1 font format and describes how to create a Type 1 font program. The book explains the specifics of the Type 1 syntax (a subset of PostScript), including information on the structure of font programs, ways to specify computer outlines, and the contents of the various font dictionaries. It also covers encryption, subroutines, and hints. http://partners.adobe.com/asn/developer/pdfs/tn/T1Format.pdf

[2] Adobe Systems Incorporated. "PostScript document structuring conventions specification (version 3.0)". Technical Note 5001, 1992.
This technical note defines a standard set of document structuring conventions (DSC), which will help ensure that a PostScript document is device independent. DSC allows PostScript language programs to communicate their document structure and printing requirements to document managers in a way that does not affect the PostScript language page description.
http://partners.adobe.com/asn/developer/pdfs/tn/5001.DSC_Spec.pdf

[3] Adobe Systems Incorporated. "Encapsulated PostScript file format specification (version 3.0)". Technical Note 5002, 1992.
This technical note details the Encapsulated PostScript file (EPSF) format, a standard format for importing and exporting PostScript language files among applications in a variety of heterogeneous environments. The EPSF format is based on and conforms to the document structuring conventions (DSC) [2].
http://partners.adobe.com/asn/developer/pdfs/tn/5002.EPSF_Spec.pdf

[4] Adobe Systems Incorporated. PostScript Language Reference. Addison-Wesley, Reading, MA, USA, 3rd edition, 1999. ISBN 0-201-37922-8.
The "red book" can be considered the definitive resource for all PostScript programmers. It contains the complete description of the PostScript language, including the latest Level 3 operators.
http://www.adobe.com/products/postscript/pdfs/PLRM.pdf

[5] Adobe Systems Incorporated. PDF Reference, version 1.4. Addison-Wesley, Boston, MA, USA, 3rd edition, 2002. ISBN 0-201-75839-3.

The specification of Adobe's Portable Document Format (PDF). The book introduces and explains all aspects of the PDF format, including its architecture and imaging model (allowing transparency and opacity for text, images, and graphics), the command syntax, the graphics operators, fonts and rendering, and the relation between PostScript and PDF. [illegible]

[6] American Mathematical Society, Providence, Rhode Island. Instructions for Preparation of Papers and Monographs: $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX, 1999.

This document contains instructions for authors preparing articles and books, using LaTeX, for publication with the American Mathematical Society (AMS) to match its publication style specifications: journals (amsart), proceedings volumes (amsproc), and monographs (amsbook). [illegible]

[7] American Mathematical Society, Providence, Rhode Island. Using the amsthm Package (Version 2.07), 2000.

The amsthm package provides an enhanced version of LaTeX's `\newtheorem` command for defining theorem-like environments, recognizing `\theoremstyle` specifications and providing a `proof` environment. [illegible]

[8] American Mathematical Society, Providence, Rhode Island. User's Guide for the amsmath Package (Version 2.0), 2002.

The amsmath package, developed by the American Mathematical Society, provides many additional features for mathematical typesetting. [illegible]

[9] American Mathematical Society, Providence, Rhode Island. User's Guide to AMSFonts Version 2.2d, 2002.

This document describes AMSFonts, the American Mathematical Society's collection of fonts of symbols and several alphabets. [illegible]

[10] J. André and Ph. Louarn. "Notes en bas de pages : comment les faire en LaTeX?" *Cahiers GUTenberg*, 12:57–70, 1991.

Several special cases of using footnotes with LaTeX are discussed—for example, how to generate a footnote referring to information inside a `tabular` or `minipage` environment, and how to reference the same footnote more than once. [illegible]

[11] Michael Barr. "A new diagram package", 2001.

A rewrite of Michael Barr's original diagram package to act as a front end to Rose's xypic (see [57, Chapter 5]). It offers a general arrow-drawing function; various common diagram shapes, such as squares, triangles, cubes, and 3×3 diagrams; small 2-arrows that can be placed anywhere in a diagram; and access to all of xypic's features. [illegible]

[12] Claudio Beccari and Apostolos Syropoulos. "New Greek fonts and the greek option of the babel package". *TUGboat*, 19(4):419–425, 1998.

Describes a new complete set of Greek fonts and their use in connection with the babel `greek` extension. [illegible]

[13] Nelson Beebe. "Bibliography prettyprinting and syntax checking". *TUGboat*, 14(4):395–419, 1993.

This article describes three software tools for BibTeX support: a pretty-printer, syntax checker, and lexical analyzer for BibTeX files; collectively called bibclean. [illegible]

[14] Barbara Beeton. "Mathematical symbols and Cyrillic fonts ready for distribution". *TUGboat*, 6(2):59–63, 1985.
The announcement of the first general release by the American Mathematical Society of the Euler series fonts. http://www.tug.org/TUGboat/Articles/tb06-2/tb11beet.pdf

[15] Frank G. Bennett, Jr. "CAMEL: kicking over the bibliographic traces in BIBTEX". *TUGboat*, 17(1):22–28, 1996.
The camel package provides a simple, logical citation interface for LATEX that allows the bibliographic style of a document to be easily changed without major editing.
http://www.tug.org/TUGboat/Articles/tb17-1/tb50benn.pdf

[16] Frank G. Bennett, Jr. "User's guide to the camel citator", 1997.
The documentation for version 1 of the camel package.
On CTAN at: macros/latex/contrib/camel

[17] A. Berdnikov, O. Lapko, M. Kolodin, A. Janishevsky, and A. Burykin. "Cyrillic encodings for LATEX 2_ε multi-language documents". *TUGboat*, 19(4):403–416, 1998.
A description of four encodings designed to support Cyrillic writing systems for the multi-language mode of LATEX 2_ε. The "raw" X2 encoding is a Cyrillic glyph container that allows one to insert into LATEX 2_ε documents text fragments written in any of the languages using a modern Cyrillic writing scheme. The T2A, T2B, and T2C encodings are genuine LATEX 2_ε encodings that may be used in a multi-language setting together with other language encodings.
http://www.tug.org/TUGboat/Articles/tb19-4/tb61berd.pdf

[18] Karl Berry. "Filenames for fonts". *TUGboat*, 11(4):517–520, 1990.
This article describes the consistent, rational scheme for font file names that was used for at least the next 15 years. Each name consists of up to eight characters (specifying the foundry, typeface name, weight, variant, expansion characteristics, and design size) that identify each font file in a unique way. http://www.tug.org/TUGboat/Articles/tb11-4/tb30berry.pdf

[19] Karl Berry. "Fontname: Filenames for TEX fonts", 2003.
The on-line documentation of the latest version of "Fontname", a scheme for TEX font file names; it explains some legal issues relating to fonts in a number of countries.
http://www.tug.org/fontname/html/index.html

[20] Javier Bezos. "The accents package", 2000.
Miscellaneous tools for mathematical accents: to create faked accents from non-accent symbols, to group accents, and to place accents below glyphs.
On CTAN at: macros/latex/contrib/bezos

[21] The *Bluebook*: A Uniform System of Citation. The Harvard Law Review Association, Cambridge, MA, 17th edition, 2000.
The *Bluebook* contains three major parts: part 1 details general standards of citation and style to be used in legal writing; part 2 presents specific rules of citation for cases, statutes, books, periodicals, foreign materials, and international materials; and part 3 consists of a series of tables showing, among other things, which authority to cite and how to abbreviate properly.
Can be ordered at: http://www.legalbluebook.com

[22] Francis Borceux. "De la construction de diagrammes". *Cahiers GUTenberg*, 5:41–48, 1990.
The diagram macros typeset diagrams consisting of arrows of different types that join at corners that can contain mathematical expressions. The macros calculate automatically the length and position of each element. The user can specify a scaling factor for each diagram.
http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/5-borceux.pdf

[23] Francis Borceux. "Diagram 3", 1993.
Commutative diagram package that uses LaTeX picture mode.
On CTAN at: macros/generic/diagrams/borceux

[24] Thierry Bouche. "Diversity in math fonts". *TUGboat*, 19(2):120–134, 1998.
Issues raised when modifying LaTeX fonts within math environments are examined. An attempt is made to suggest effective means of accessing a larger variety of font options, while avoiding typographic nonsense. http://www.tug.org/TUGboat/Articles/tb19-2/tb59bouc.pdf

[25] Johannes Braams. "Babel, a multilingual style-option system for use with LaTeX's standard document styles". *TUGboat*, 12(2):291–301, 1991.
The babel package was originally a collection of document-style options to support different languages. An update was published in *TUGboat*, 14(1):60–62, April 1993.
http://www.tug.org/TUGboat/Articles/tb12-2/tb32braa.pdf
http://www.tug.org/TUGboat/Articles/tb14-1/tb38braa.pdf

[26] Neil Bradley. The XML Companion. Addison-Wesley, Boston, MA, USA, 3rd edition, 2002. ISBN 0-201-77059-8.
This book provides a description of XML features without assuming knowledge of HTML or SGML, covering also related standards such as Xpath, XML Schema, SAX, DOM, XSLT, Xlink, and Xpointer.

[27] Peter Breitenlohner et al. "The eTeX manual (version 2)", 1998.
The current manual for the eTeX system, which extends the capabilities of TeX while retaining compatibility. On CTAN at: systems/e-tex/v2/doc/etex_man.pdf

[28] Robert Bringhurst. The elements of typographic style. Hartley & Marks Publishers, Point Roberts, WA, USA, and Vancouver, BC, Canada, 2nd edition, 1996. ISBN 0-88179-133-4 (hardcover), 0-88179-132-6 (paperback).
A very well-written book on typography with a focus on the proper use of typefaces.

[29] Judith Butcher. Copy-editing: The Cambridge handbook for editors, authors and publishers. Cambridge University Press, New York, 3rd edition, 1992. ISBN 0-521-40074-0.
A reference guide for all those involved in the process of preparing typescripts and illustrations for printing and publication. The book covers all aspects of the editorial process, from the basics of how to mark a typescript for the designer and the typesetter, through the ground rules of house style and consistency, to how to read and correct proofs.

[30] David Carlisle. "A LaTeX tour, Part 1: The basic distribution". *TUGboat*, 17(1):67–73, 1996.
A "guided tour" around the files in the basic LaTeX distribution. File names and paths relate to the file hierarchy of the CTAN archives.
http://www.tug.org/TUGboat/Articles/tb17-1/tb50carl.pdf

[31] David Carlisle. "A LaTeX tour, Part 2: The tools and graphics distributions". *TUGboat*, 17(3):321–326, 1996.
A "guided tour" around the "tools" and "graphics" packages. Note that *The Manual* [104] assumes that at least the graphics distribution is available with standard LaTeX.
http://www.tug.org/TUGboat/Articles/tb17-3/tb52carl.pdf

[32] David Carlisle. "A LaTeX tour, Part 3: mfnfss, psnfss and babel". *TUGboat*, 18(1):48–55, 1997.
A "guided tour" through three more distributions that are part of the standard LaTeX system. The mfnfss distribution provides LaTeX support for some popular METAFONT-produced fonts

that do not otherwise have any LaTeX interface. The psnfss distribution consists of LaTeX packages giving access to PostScript fonts. The babel distribution provides LaTeX with multilingual capabilities. http://www.tug.org/TUGboat/Articles/tb18-1/tb54carl.pdf

[33] David Carlisle. "OpenMath, MathML, and XSL". *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, 34(2):6–11, 2000.
Discussion of XML markup for mathematics—in particular, OpenMath and MathML—and the use of XSLT to transform between these languages.
Restricted to ACM members; http://www.acm.org/sigsam/bulletin/issues/issue132.html

[34] David Carlisle. "XMLTEX: A non validating (and not 100% conforming) namespace aware XML parser implemented in TeX". *TUGboat*, 21(3):193–199, 2000.
XMLTEX is a an XML parser and typesetter implemented in TeX, which by default uses the LaTeX kernel to provide typesetting functionality.
http://www.tug.org/TUGboat/Articles/tb21-3/tb68carl.pdf

[35] David Carlisle, Patrick Ion, Robert Miner, and Nico Poppelier, editors. Mathematical Markup Language (MathML) Version 2.0. W3C, 2nd edition, 2003.
MathML is an XML vocabulary for mathematics, designed for use in browsers and as a communication language between computer algebra systems. http://www.w3.org/TR/MathML2

[36] David Carlisle, Chris Rowley, and Frank Mittelbach. "The LaTeX3 Programming Language—a proposed system for TeX macro programming". *TUGboat*, 18(4):303–308, 1997.
Some proposals for a radically new syntax and software tools.
http://www.tug.org/TUGboat/Articles/tb18-4/tb57rowl.pdf

[37] Pehong Chen and Michael A. Harrison. "Index preparation and processing". *Software—Practice and Experience*, 19(9):897–915, 1988.
A description of the makeindex system.

[38] The Chicago Manual of Style. University of Chicago Press, Chicago, IL, USA, 15th edition, 2003. ISBN 0-226-10403-6.
The standard U.S. publishing style reference for authors and editors.

[39] Adrian F. Clark. "Practical halftoning with TeX". *TUGboat*, 12(1):157–165, 1991.
Reviews practical problems encountered when using TeX for typesetting half-tone pictures and compares other techniques to include graphics material. Advantages and disadvantages of the various approaches are described and some attempts at producing color separations are discussed. http://www.tug.org/TUGboat/Articles/tb12-1/tb31clark.pdf

[40] Matthias Clasen and Ulrik Vieth. "Towards a new math font encoding for (LA)TeX". *Cahiers GUTenberg*, 28-29:94–121, 1998.
A prototype implementation of 8-bit math font encodings for LaTeX.
http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-clasen.pdf

[41] Carl Dair. Design with Type. University of Toronto Press, Toronto, Ontario, Canada, 1967. ISBN 0-8020-1426-7 (hardcover), 0-8020-6519-8 (paperback).
A good survey of traditional typography with many useful rules of thumb.

[42] Michael Downes. "Breaking equations". *TUGboat*, 18(3):182–194, 1997.
TeX is not very good at displaying equations that must be broken into more than one line. The breqn package eliminates many of the most significant problems by supporting automatic line breaking of displayed equations.
http://www.tug.org/TUGboat/Articles/tb18-3/tb56down.pdf

[43] Michael Downes. "The amsrefs LaTeX package and the amsxport BibTeX style". *TUGboat*, 21(3):201–209, 2000.
Bibliography entries using the amsrefs format provide a rich internal structure and high-level markup close to that traditionally found in BibTeX database files. On top of that, using amsrefs markup lets you specify the bibliography style completely in a LaTeX document class file.
http://www.tug.org/TUGboat/Articles/tb21-3/tb68down.pdf

[44] Dudenredaktion, editor. Duden, Rechtschreibung der deutschen Sprache. Dudenverlag, Mannheim, 21st edition, 1996. ISBN 3-411-04011-4.
The standard reference for the correct spelling of all words of contemporary German and for hyphenation rules, with examples and explanations for difficult cases, and a comparison of the old and new orthographic rules.

[45] Victor Eijkhout. TeX by Topic, A TeXnician's Reference. Addison-Wesley, Reading, MA, USA, 1991. ISBN 0-201-56882-9. Out of print. Available free of charge from the author in PDF format.
A systematic reference manual for the experienced TeX user. The book offers a comprehensive treatment of every aspect of TeX, with detailed explanations of the mechanisms underlying TeX's working, as well as numerous examples of TeX programming techniques.
http://www.eijkhout.net/tbt

[46] Robin Fairbairns. "UK list of TeX frequently asked questions on the Web", 2003.
This list of Frequently Asked Questions on TeX was originated by the Committee of the U.K. TeX Users' Group; it has well over 300 entries and is regularly updated and expanded.
http://www.tex.ac.uk/faq

[47] Laurence Finston. "Spindex—Indexing with special characters". *TUGboat*, 18(4):255–273, 1997.
Common Lisp indexing program and supporting TeX macros for indexes that include non-Latin characters. http://www.tug.org/TUGboat/Articles/tb18-4/tb57fins.pdf

[48] Shinsaku Fujita and Nobuya Tanaka. "XΥMTeX (Version 2.00) as implementation of the XΥM notation and the XΥM markup language". *TUGboat*, 21(1):7–14, 2000.
A description of version 2 of the XΥMTeX system, which can be regarded as a linear notation system expressed in TeX macros that corresponds to the IUPAC (International Union of Pure and Applied Chemistry) nomenclature. It provides a convenient method for drawing complicated structural formulas. http://www.tug.org/TUGboat/Articles/tb21-1/tb66fuji.pdf

[49] Shinsaku Fujita and Nobuya Tanaka. "Size reduction of chemical structural formulas in XΥMTeX (Version 3.00)". *TUGboat*, 22(4):285–289, 2001.
Further improvements to the XΥMTeX system, in particular in the area of size reduction of structural formulas. http://www.tug.org/TUGboat/Articles/tb22-4/tb72fuji.pdf

[50] Rei Fukui. "TIPA: A system for processing phonetic symbols in LaTeX". *TUGboat*, 17(2):102–114, 1996.
TIPA is a system for processing symbols of the International Phonetic Alphabet with LaTeX. It introduces a new encoding for phonetic symbols (T3), which includes all the symbols and diacritics found in the recent versions of IPA as well as some non-IPA symbols. It has full support

for LaTeX 2ε and offers an easy input method in the IPA environment.
http://www.tug.org/TUGboat/Articles/tb17-2/tb51rei.pdf

[51] Bernard Gaulle. "Comment peut-on personnaliser l'extension french de LaTeX?" *Cahiers GUTenberg*, 28-29:143-157, 1998.
Describes how to personalize the french package.
http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-gaulle.pdf

[52] Maarten Gelderman. "A short introduction to font characteristics". *TUGboat*, 20(2):96-104, 1999.
This paper provides a description of the main aspects used to describe a font, its basic characteristics, elementary numerical dimensions to access properties of a typeface design, and the notion of "contrast". http://www.tug.org/TUGboat/Articles/tb20-2/tb63geld.pdf

[53] Charles F. Goldfarb. The SGML Handbook. Oxford University Press, London, Oxford, New York, 1990. ISBN 0-19-853737-9.
The full text of the ISO SGML standard [68] copiously annotated by its author, and several tutorials.

[54] Norbert Golluch. Kleinweich Büro auf Schlabberscheiben. Eichborn, Frankfurt, 1999.
Tecknisches Deutsch für Angefangen.

[55] Michel Goossens, Frank Mittelbach, and Alexander Samarin. The LaTeX Companion. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, 1994. ISBN 0-201-54199-8.
The first edition of this book.

[56] Michel Goossens and Sebastian Rahtz. The LaTeX Web Companion: Integrating TeX, HTML, and XML. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1999. ISBN 0-201-43311-7. With Eitan M. Gurari, Ross Moore, and Robert S. Sutor.
This book teaches (scientific) authors how to publish on the web or other hypertext presentation systems, building on their experience with LaTeX and taking into account their specific needs in fields such as mathematics, non-European languages, and algorithmic graphics. The book explains how to make full use of the Adobe Acrobat format from LaTeX, convert legacy documents to HTML or XML, make use of math in web applications, use LaTeX as a tool in preparing web pages, read and write simple XML/SGML, and produce high-quality printed pages from web-hosted XML or HTML pages using TeX or PDF.

[57] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. The LaTeX Graphics Companion: Illustrating Documents with TeX and PostScript. Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, 1997. ISBN 0-201-85469-4.
The book shows how to incorporate graphic files into a LaTeX document, program technical diagrams using several different languages, produce color pictures, achieve special effects with fragments of embedded PostScript, and make high-quality music scores and game diagrams. It also contains detailed descriptions of important packages such as xypic, pstricks, and MetaPost, the standard LaTeX color and graphics packages, PostScript fonts and how to use them in LaTeX, and the dvips and ghostscript programs.

[58] Michel Goossens and Vesa Sivunen. "LaTeX, SVG, Fonts". *TUGboat*, 22(4):269-279, 2001.
A short overview of SVG and its advantages for portable graphics content, conversion of PostScript glyph outlines to SVG outlines, and the use of SVG glyphs in TeX documents.
http://www.tug.org/TUGboat/Articles/tb22-4/tb72goos.pdf

[59] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. Concrete Mathematics. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-55802-5.

A mathematics textbook prepared with TEX using the Concrete Roman typeface; see also [92].

[60] George Grätzer. Math into LATEX. Birkhäuser and Springer-Verlag, Cambridge, MA, USA; Berlin, Germany/Basel, Switzerland, and Berlin, Germany/Heidelberg, Germany/London, UK/ etc., 3rd edition, 2000. ISBN 0-8176-4131-9, 3-7643-4131-9.

Provides a general introduction to LATEX as used to prepare mathematical books and articles. Covers AMS document classes and packages in addition to the basic LATEX offerings.

[61] George D. Greenwade. "The Comprehensive TEX Archive Network (CTAN)". *TUGboat*, 14(3):342–351, 1993.

An outline of the conception, development, and early use of the CTAN archive, which makes all TEX-related files available on the network.

`http://www.tug.org/TUGboat/Articles/tb14-3/tb40green.pdf`

[62] Yannis Haralambous. "Typesetting old German: Fraktur, Schwabacher, Gotisch and initials". *TUGboat*, 12(1):129–138, 1991.

Demonstrates the use of METAFONT to recreate faithful copies of old-style typefaces and explains the rules for typesetting using these types, with examples.

`http://www.tug.org/TUGboat/Articles/tb12-1/tb31hara.pdf`

[63] Horace Hart. Hart's Rules; For Compositors and Readers at the University Press, Oxford. Oxford University Press, London, Oxford, New York, 39th edition, 1991. ISBN 0-19-212983-X.

A widely used U.K. reference for authors and editors. With the *Oxford Dictionary for Writers and Editors* it presents the canonical house style of the Oxford University Press. See also [143].

[64] Alan Hoenig. TEX Unbound: LATEX and TEX Strategies for Fonts, Graphics, & More. Oxford University Press, London, Oxford, New York, 1998. ISBN 0-19-509686-X (paperback), 0-19-509685-1 (hardcover).

The first part of this book provides a brief but comprehensive overview of TEX, LATEX, METAFONT, and MetaPost, with particular emphasis on how everything fits together, how the production cycle works, and what kinds of files are involved. The second part is devoted to details of fonts and their use in TEX. Of particular interest are 30 pages of examples showing how various combinations of well-known text typefaces might be used together with the few choices of math fonts currently available. The final part of the book discusses graphics applications—in particular, TEX-friendly methods such as METAFONT and MetaPost, the pstricks package, PICTEX, and MFpic.

[65] Berthold K. P. Horn. "The European Modern fonts". *TUGboat*, 19(1):62–63, 1998.

The European Modern (EM) fonts are Type 1 fonts based on Computer Modern (CM) that have ready-made accented and composite characters, thus enabling TEX hyphenation when using languages that use such characters.

`http://www.tug.org/TUGboat/Articles/tb19-1/tb58horn.pdf`

[66] Jean-Michel Hufflen. "Typographie: les conventions, la tradition, les goûts,..., et LATEX". *Cahiers GUTenberg*, 35–36:169–214, 2000.

This article shows that learning typographic rules—even considering those for French and English together—is not all that difficult. It also teaches the basics of using the LATEX packages french (for French only) and babel (allowing a homogeneous treatment of most other languages).

Finally, the author shows how to build a new multilingual document class and bibliography style.
`http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/35-hufflen.pdf`

[67] "ISO/IEC 8859-1:1998 to ISO/IEC 8859-16:2001, Information technology—8-bit single-byte coded graphic character sets, Parts 1 to 16". International Standard ISO/IEC 8859, ISO Geneva, 1998-2001.
A description of various 8-bit alphabetic character sets. Parts 1-4, 9, 10, and 13-16 correspond to 10 character sets needed to encode different groups of languages using the Latin alphabet, while part 5 corresponds to Cyrillic, part 6 to Arabic, part 7 to Greek, part 8 to Hebrew, and part 11 to Thai.

[68] "ISO 8879:1986, Information Processing—Text and Office Systems—Standard Generalised Markup Language (SGML)". International Standard ISO 8879, ISO Geneva, 1986.
The—not always easy to read—ISO standard describing the SGML language in full technical detail. An addendum was published in 1988 and two corrigenda in 1996 and 1999. See [53] for an annotated description.

[69] "ISO/IEC 10646-1:2000, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane". International Standard ISO 10646-1 (Edition 2), ISO Geneva, 2000.
This standard specifies the architecture of the Universal Multiple-Octet Coded Character Set (UCS). This 32-bit character encoding standard is for all practical purposes identical to the Unicode standard; see [165]. The layout of the Basic Multilingual Plane (plane 0 or BMP) is described in detail. An amendment in 2002 added mathematical symbols and other characters.

[70] "ISO/IEC 10646-2:2001, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 2: Supplementary Planes". International Standard ISO 10646-2, ISO Geneva, 2001.
Complementing [69], which describes plane 0 (BMP) of the UCS, the present standard details the layout of the supplementary planes; see also [165].

[71] "ISO/IEC 14651:2001, Information technology—International string ordering and comparison—Method for comparing character strings and description of the common template tailorable ordering". International Standard ISO/IEC 14651:2001, ISO Geneva, 2001.

[72] Alan Jeffrey. "PostScript font support in LaTeX 2ε". *TUGboat*, 15(3):263-268, 1994.
Describes the original psnfss distribution for using PostScript fonts with LaTeX.
`http://www.tug.org/TUGboat/Articles/tb15-3/tb44jeff.pdf`

[73] Alan Jeffrey. "Tight setting with TeX". *TUGboat*, 16(1):78-80, 1995.
Describes some experiments with setting text matter in TeX using Adobe Times, a very tightly spaced text font. `http://www.tug.org/TUGboat/Articles/tb16-1/tb46jeff.pdf`

[74] Alan Jeffrey and Rowland McDonnell. "fontinst: Font installation software for TeX", 1998.
This utility package supports the creation of complex virtual fonts in any encoding for use with LaTeX, particularly from collections of PostScript fonts.
On CTAN at: `fonts/utilities/fontinst/doc/manual`

[75] Alan Jeffrey, Sebastian Rahtz, Ulrik Vieth, and Lars Hellström. "The fontinst utility", 2003.
Technical description of the fontinst utility.
On CTAN at: `fonts/utilities/fontinst/source/fisource.dvi`

[76] Roger Kehr. "xindy—A flexible indexing system". *Cahiers GUTenberg*, 28-29:223-230, 1998.

A new index processor, xindy, is described. It allows for sorting of index entries at a fine granularity in a multi-language environment, offers new mechanisms for processing structured location references besides page numbers and Roman numerals, and has provisions for complex markup schemes.
http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-kehr.pdf

[77] Brian W. Kernighan. "pic—A graphics language for typesetting". Computing Science Technical Report 116, AT&T Bell Laboratories, 1991.

The user manual for the pic language, which is intended for drawing simple figures on a typesetter. The basic objects of the language are boxes, circles, ellipses, lines, arrows, spline curves, and text. These may be placed at any position, specified either in an absolute way or with respect to previous objects. http://cm.bell-labs.com/cm/cs/cstr/116.ps.gz

[78] Jörg Knappen. "Release 1.2 of the dc-fonts: Improvements to the European letters and first release of text companion symbols". *TUGboat*, 16(4):381-387, 1995.

Description of the DC fonts, which were precursors of the EC fonts, which themselves are the default fonts for the T1 encoding of LaTeX.
http://www.tug.org/TUGboat/Articles/tb16-4/tb49knap.pdf

[79] Jörg Knappen. "The dc fonts 1.3: Move towards stability and completeness". *TUGboat*, 17(2):99-101, 1996.

A follow-up article to [78]. It explains the progress made in version 1.3 in the areas of stability and completeness. http://www.tug.org/TUGboat/Articles/tb17-2/tb51knap.pdf

[80] Donald E. Knuth. TeX and METAFONT—New Directions in Typesetting. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, 1979. ISBN 0-932376-02-9.

Contains an article on "Mathematical Typography", describing the author's motivation for starting to work on TeX and the early history of computer typesetting. Describes early (now obsolete) versions of TeX and METAFONT.

[81] Donald E. Knuth. "Literate programming". Report STAN-CS-83-981, Stanford University, Department of Computer Science, Stanford, CA, USA, 1983.

A collection of papers on styles of programming and documentation.
http://www.literateprogramming.com/farticles.html.

[82] Donald E. Knuth. The TeXbook, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0.

The definitive user's guide and complete reference manual for TeX.

[83] Donald E. Knuth. TeX: The Program, volume B of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13437-3.

The complete source code for the TeX program, typeset with several indices.

[84] Donald E. Knuth. The METAFONTbook, volume C of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13445-4 (hardcover), 0-201-13444-6 (paperback).

The user's guide and reference manual for METAFONT, the companion program to TeX for designing fonts.

[85] Donald E. Knuth. METAFONT: The Program, volume D of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13438-1.
The complete source code listing of the METAFONT program.

[86] Donald E. Knuth. Computer Modern Typefaces, volume E of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13446-2.
More than 500 Greek and Roman letterforms, together with punctuation marks, numerals, and many mathematical symbols, are graphically depicted. The METAFONT code to generate each glyph is given and it is explained how, by changing the parameters in the METAFONT code, all characters in the Computer Modern family of typefaces can be obtained.

[87] Donald E. Knuth. 3:16 Bible texts illuminated. A-R Editions, Inc., Madison, Wisconsin, 1990. ISBN 0-89579-252-4.
Analysis of Chapter 3 Verse 16 of each book of the Bible. Contains wonderful calligraphy.

[88] Donald E. Knuth. The Art of Computer Programming, vols 1–3. Addison-Wesley, Reading, MA, USA, 1998. ISBN 0-201-89683-4, 0-201-03822-6, 0-201-03803-X.
A major work on algorithms and data structures for efficient programming.

[89] Donald E. Knuth. Digital Typography. CSLI Publications, Stanford, CA, USA, 1999. ISBN 1-57586-011-2 (cloth), 1-57586-010-4 (paperback).
A collection of Knuth's writings on TeX and typography.

[90] Donald E. Knuth. "Mathematical typography". In Knuth [89], pp. 19–65.
Based on a lecture he gave in 1978, Knuth makes the point that mathematics books and journals do not look as beautiful now as they did in the past. As this is mainly due to the fact that high-quality typesetting has become too expensive, he proposes to use mathematics itself to solve the problem. As a first step he sees the development of a method to unambiguously mark up the math elements in a document so that they can be easily handled by machines. The second step is to use mathematics to design the shapes of letters and symbols. The article goes into the details of these two approaches.

[91] Donald E. Knuth. "Virtual fonts: More fun for grand wizards". In Knuth [89], pp. 247–262. Originally published in *TUGboat* 11(1):13–23, 1990.
An explanation of what virtual fonts are and why they are needed, plus technical details.
On CTAN at: `info/virtual-fonts.knuth`
`http://www.tug.org/TUGboat/Articles/tb11-1/tb27knut.pdf`

[92] Donald E. Knuth. "Typesetting concrete mathematics". In Knuth [89], pp. 367–378. Originally published in *TUGboat* 10(1):31–36, 1989.
Knuth explains how he prepared the textbook *Concrete Mathematics*. He states that he wanted to make that book both mathematically and typographically "interesting", since it would be the first major use of Herman Zapf's new typeface, AMS Euler. The font parameters were tuned up to make the text look as good as that produced by the best handwriting of a mathematician. Other design decisions for the book are also described.
`http://www.tug.org/TUGboat/Articles/tb10-1/tb26knut.pdf`

[93] Donald E. Knuth. "Fonts for digital halftones". In Knuth [89], pp. 415–448. Originally published in *TUGboat* 8(2):135–160, 1987.
This article discusses some experiments in which METAFONT was used to create fonts to generate half-tones on laser printers. The methods also proved useful in several other applications, while their design involved a number of interesting issues.
`http://www.tug.org/TUGboat/Articles/tb08-2/tb18knut.pdf`

[94] Donald E. Knuth. "Computers and typesetting". In Knuth [89], pp. 555–562. Originally published in *TUGboat* 7(2):95–98, 1986.

Remarks presented by Knuth at the Computer Museum, Boston, Massachusetts, on 21 May 1986, at the "coming-out" party to celebrate the completion of TeX.

http://www.tug.org/TUGboat/Articles/tb07-2/tb14knut.pdf

[95] Donald E. Knuth. "The new versions of TeX and METAFONT". In Knuth [89], pp. 563–570. Originally published in *TUGboat* 10(3):325–328, 1989.

Knuth explains how he was convinced at the TUG Meeting at Stanford in 1989 to make one further set of changes to TeX and METAFONT to extend these programs to support 8-bit character sets. He goes on to describe the various changes he introduced to implement this feature, as well as a few other improvements.

http://www.tug.org/TUGboat/Articles/tb10-3/tb25knut.pdf

[96] Donald E. Knuth. "The future of TeX and METAFONT". In Knuth [89], pp. 571–572. Originally published in *TUGboat* 11(4):489, 1990.

In this article Knuth announces that his work on TeX, METAFONT, and Computer Modern has "come to an end" and that he will make further changes only to correct extremely-serious bugs.

http://www.tug.org/TUGboat/Articles/tb11-4/tb30knut.pdf

[97] Donald E. Knuth and Pierre MacKay. "Mixing right-to-left texts with left-to-right texts". In Knuth [89], pp. 157–176. Originally published in *TUGboat* 8(1):14–25, 1987.

TeX was initially designed to produce documents with material flowing left-to-right and top-to-bottom. This paper clarifies the issues involved in mixed-direction document production and discusses changes to TeX that can extend it to become a bidirectional formatting system.

http://www.tug.org/TUGboat/Articles/tb08-1/tb17knutmix.pdf

[98] Donald E. Knuth and Michael F. Plass. "Breaking paragraphs into lines". In Knuth [89], pp. 67–155.

This article, originally published in 1981, addresses the problem of dividing the text of a paragraph into lines of approximately equal length. The basic algorithm considers the paragraph as a whole and introduces the (now well-known TeX) concepts of "boxes", "glue", and "penalties" to find optimal breakpoints for the lines. The paper describes the dynamic programming technique used to implement the algorithm.

[99] Donald E. Knuth and Hermann Zapf. "AMS Euler—A new typeface for mathematics". In Knuth [89], pp. 339–366.

The two authors explain, in this article originally published in 1989, how a collaboration between scientists and artists is helping to bring beauty to the pages of mathematical journals and textbooks.

[100] Markus Kohm and Jens-Uwe Morawski. KOMA-Script: eine Sammlung von Klassen und Paketen für LaTeX 2ε. DANTE, Heidelberg, 2003. ISBN 3-936427-45-3.

KOMA·Script is a bundle of LaTeX classes and packages that can be used as replacements for the standard LaTeX classes offering extended functionalities. German and English manuals are provided as part of the distribution.

On CTAN at: macros/latex/contrib/koma-script/scrguide.pdf

[101] Helmut Kopka and Patrick Daly. Guide to LaTeX. Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, MA, USA, 4th edition, 2004. ISBN 0-201-17385-6.

An introductory guide to LaTeX with a different pedagogical style than Lamport's *LaTeX Manual* [104].

[102] Klaus Lagally. "ArabTEX—Typesetting Arabic with vowels and ligatures". In "Proceedings of the 7th European TEX Conference, Prague", pp. 153–172. CsTUG, Prague, 1992. ISBN 80-210-0480-0.

A macro package, compatible with plain TEX and LATEX, for typesetting Arabic with both partial and full vocalization.

[103] Leslie Lamport. "*MakeIndex*, An Index Processor For LATEX". Technical report, Electronic Document in *MakeIndex* distribution, 1987.

This document explains the syntax that can be used inside LATEX's `\index` command when using *MakeIndex* to generate your index. It also gives a list of the possible error messages.
On CTAN at: `indexing/makeindex/doc/makeindex.dvi`

[104] Leslie Lamport. LATEX: A Document Preparation System: User's Guide and Reference Manual. Addison-Wesley, Reading, MA, USA, 2nd edition, 1994. ISBN 0-201-52983-1. Reprinted with corrections in 1996.

The ultimate reference for basic user-level LATEX by the creator of LATEX 2.09. It complements the material presented in this book.

[105] Olga Lapko and Irina Makhovaya. "The style `russianb` for Babel: Problems and solutions". *TUGboat*, 16(4):364–372, 1995.

This paper describes the language option `russianb`, which includes specific commands to russify captions and alphabetic counters and to allow for Russian mathematical operators. Some problems are mentioned that may occur when using this option (i.e., with different encodings).
`http://www.tug.org/TUGboat/Articles/tb16-4/tb49olga.pdf`

[106] LATEX3 Project Team. "LATEX bug database".

The bug reporting and tracking service run by the LATEX3 team as part of the LATEX 2_ε maintenance activity. `http://www.latex-project.org/cgi-bin/ltxbugs2html`

[107] LATEX3 Project Team. "LATEX news".

An issue of *LATEX News* is released with each LATEX 2_ε release, highlighting changes since the last release. `http://www.latex-project.org/ltnews/`

[108] LATEX3 Project Team. "Default docstrip headers". *TUGboat*, 19(2):137–138, 1998.

This document describes the format of the header that docstrip normally adds to generated package files. This header is suitable for copyright information or distribution conditions.
`http://www.tug.org/TUGboat/Articles/tb19-2/tb59ltdocstrip.pdf`

[109] LATEX3 Project Team. "LATEX 2_ε font selection", 2000.

A description of font selection in standard LATEX intended for package writers who are already familiar with TEX fonts and LATEX. `http://www.latex-project.org/guides/fntguide.pdf`

[110] LATEX3 Project Team. "Configuration options for LATEX 2_ε", 2001.

How to configure a LATEX installation using the set of standard configuration files.
`http://www.latex-project.org/guides/cfgguide.pdf`

[111] LATEX3 Project Team. "The LATEX project public license (version 1.3)", 2003.

An Open Source License used by the core LATEX 2_ε distribution and many contributed packages.
`http://www.latex-project.org/lppl/`

[112] John Lavagnino and Dominik Wujastyk. "An overview of EDMAC: A plain TEX format for critical editions". *TUGboat*, 11(4):623–643, 1990.

EDMAC is for typesetting of "critical editions" of texts such as the *Oxford Classical Texts, Shakespeare*, and other series. It supports marginal line numbering and multiple series of footnotes and endnotes keyed to line numbers.
`http://www.tug.org/TUGboat/Articles/tb11-4/tb30lava.pdf`

[113] Werner Lemberg. "The CJK package: Multilingual support beyond Babel". *TUGboat*, 18(3):214–224, 1997.
A description of the CJK (Chinese/Japanese/Korean) package for LaTeX and its interface to mule (multilingual emacs). `http://www.tug.org/TUGboat/Articles/tb18-3/cjkintro600.pdf`

[114] Silvio Levy. "Using Greek fonts with TeX". *TUGboat*, 9(1):20–24, 1988.
The author tries to demonstrate that typesetting Greek in TeX with the gr family of fonts can be as easy as typesetting English text and leads to equally good results. The article is meant as a tutorial but some technical details are given for those who will have acquired greater familiarity with the font. `http://www.tug.org/TUGboat/Articles/tb09-1/tb20levy.pdf`

[115] Franklin Mark Liang. Word Hy-phen-a-tion by Com-pu-ter. Ph.D. thesis, Stanford University, Stanford, CA 94305, 1983. Also available as Stanford University, Department of Computer Science Report No. STAN-CS-83-977.
A detailed description of the word hyphenation algorithm used by TeX.

[116] Ruari McLean. The Thames and Hudson Manual of Typography. Thames and Hudson, London, UK, 1980. ISBN 0-500-68022-1.
A broad introduction to traditional commercial typography.

[117] Frank Mittelbach. "E-TeX: Guidelines for future TeX". *TUGboat*, 11(3):337–345, 1990.
The output of TeX is compared with that of hand-typeset documents. It is shown that many important concepts of high-quality typesetting are not supported and that further research to design a "successor" typesetting system to TeX should be undertaken.
`http://www.tug.org/TUGboat/Articles/tb11-3/tb29mitt.pdf`

[118] Frank Mittelbach. "Comments on "Filenames for Fonts" (*TUGboat* 11#4)". *TUGboat*, 13(1):51–53, 1992.
Some problems with K. Berry's naming scheme are discussed, especially from the point of view of defining certain font characteristics independently and the use of the scheme with NFSS.
`http://www.tug.org/TUGboat/Articles/tb13-1/tb34mittfont.pdf`

[119] Frank Mittelbach. "A regression test suite for LaTeX 2ε". *TUGboat*, 18(4):309–311, 1997.
Description of the concepts and implementation of the test suite used to test for unexpected side effects after changes to the LaTeX kernel. One of the most valuable maintenance tools for keeping LaTeX 2ε stable. `http://www.tug.org/TUGboat/Articles/tb18-4/tb57mitt.pdf`

[120] Frank Mittelbach. "Language Information in Structured Documents: Markup and rendering—Concepts and problems". In "International Symposium on Multilingual Information Processing", pp. 93–104. Tsukuba, Japan, 1997. Invited paper. Republished in *TUGboat* 18(3):199–205, 1997.
This paper discusses the structure and processing of multilingual documents, both at a general level and in relation to a proposed extension to standard LaTeX.
`http://www.tug.org/TUGboat/Articles/tb18-3/tb56lang.pdf`

[121] Frank Mittelbach. "Formatting documents with floats: A new algorithm for LaTeX 2ε". *TUGboat*, 21(3):278–290, 2000.
Descriptions of features and concepts of a new output routine for LaTeX that can handle spanning floats in multicolumn page design.
`http://www.tug.org/TUGboat/Articles/tb21-3/tb68mittel.pdf`

[122] Frank Mittelbach. "The trace package". *TUGboat*, 22(1/2):93–99, 2001.
A description of the trace package for controlling debugging messages from LaTeX packages.
`http://www.tug.org/TUGboat/Articles/tb22-1-2/tb70mitt.pdf`

[123] Frank Mittelbach, David Carlisle, and Chris Rowley. "New interfaces for LaTeX class design, Parts I and II". *TUGboat*, 20(3):214–216, 1999.
Some proposals for the first-ever interface to setting up and coding LaTeX classes.
`http://www.tug.org/TUGboat/Articles/tb20-3/tb64carl.pdf`

[124] Frank Mittelbach, David Carlisle, Chris Rowley, et al. "Experimental LaTeX code for class design".
At the TeX Users Group conference in Vancouver the LaTeX project team gave a talk on models for user-level interfaces and designer-level interfaces in LaTeX3 [123]. Most of these ideas have been implemented in prototype implementations (e.g., template design, front matter handling, output routine, galley and paragraph formatting). The source code is documented and contains further explanations and examples; see also [121].
Slides: `http://www.latex-project.org/papers/tug99pdf`
Code: `http://www.latex-project.org/code/experimental`

[125] Frank Mittelbach, Denys Duchier, Johannes Braams, Marcin Woliński, and Mark Wooding. "The docstrip program", 2003. Distributed as part of the base LaTeX distribution.
Describes the implementation of the docstrip program.
On CTAN at: `macros/latex/base/docstrip.dtx`

[126] Frank Mittelbach and Chris Rowley. "LaTeX 2.09 $\hookrightarrow$ LaTeX3". *TUGboat*, 13(1):96–101, 1992.
A brief sketch of the LaTeX3 Project, retracing its history and describing the structure of the system. An update appeared in *TUGboat*, 13(3):390–391, October 1992. A call for volunteers to help in the development of LaTeX3 and a list of the various tasks appeared in *TUGboat*, 13(4):510–515, December 1992. The article also describes how you can obtain the current task list as well as various LaTeX3 working group documents via e-mail or FTP and explains how you can subscribe to the LaTeX3 discussion list.
`http://www.tug.org/TUGboat/Articles/tb13-1/tb34mittl3.pdf`

[127] Frank Mittelbach and Chris Rowley. "The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography?" In C. Vanoirbeek and G. Coray, editors, "EP92—Proceedings of Electronic Publishing, '92, International Conference on Electronic Publishing, Document Manipulation, and Typography, Swiss Federal Institute of Technology, Lausanne, Switzerland, April 7-10, 1992", pp. 261–273. Cambridge University Press, New York, 1992. ISBN 0-521-43277-4.

[128] Frank Mittelbach and Rainer Schöpf. "A new font selection scheme for TeX macro packages—the basic macros". *TUGboat*, 10(2):222–238, 1989.
A description of the basic macros used to implement the first version of LaTeX's New Font Selection Scheme. `http://www.tug.org/TUGboat/Articles/tb10-2/tb24mitt.pdf`

[129] Frank Mittelbach and Rainer Schöpf. "With LaTeX into the nineties". *TUGboat*, 10(4):681–690, 1989.
This article proposes a reimplementation of LaTeX that preserves the essential features of the current interface while taking into account the increasing needs of the various user communities. It also formulates some ideas for further developments. It was instrumental in the move from LaTeX 2.09 to LaTeX 2_ε. `http://www.tug.org/TUGboat/Articles/tb10-4/tb26mitt.pdf`

[130] Frank Mittelbach and Rainer Schöpf. "Reprint: The new font family selection — User interface to standard LaTeX". *TUGboat*, 11(2):297–305, 1990.
A complete description of the user interface of the first version of LaTeX's New Font Selection Scheme. `http://www.tug.org/TUGboat/Articles/tb11-2/tb28mitt.pdf`

[131] Frank Mittelbach and Rainer Schöpf. "Towards LaTeX 3.0". *TUGboat*, 12(1):74–79, 1991.
The objectives of the LaTeX3 project are described. The authors examine enhancements to LaTeX's user and style file interfaces that are necessary to keep pace with modern developments, such as SGML. They also review some internal concepts that need revision.
http://www.tug.org/TUGboat/Articles/tb12-1/tb31mitt.pdf

[132] Gerd Neugebauer. "BIBTOOL: A tool to manipulate BibTeX files", 2002.
Describes the bibtool program for pretty-printing, sorting and merging of BibTeX databases, generation of uniform reference keys, and selecting of references used in a publication.
On CTAN at: biblio/bibtex/utils/bibtool/bibtool.dvi

[133] O. Nicole, J. André, and B. Gaulle. "Notes en bas de pages : commentaires". *Cahiers GUTenberg*, 15:46–32, 1993.
Comments, clarifications, and additions to [10].
http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/15-nicole.pdf

[134] Scott Pakin. "The comprehensive LaTeX symbol list", 2003.
This document lists more than 2800 symbols and the corresponding LaTeX commands that produce them. Some of these symbols are guaranteed to be available in every LaTeX 2_ε system; others require fonts and packages that may not accompany a given distribution and that therefore need to be installed. All of the fonts and packages described in the document are freely available from the CTAN archives. On CTAN at: info/symbols/comprehensive/

[135] Oren Patashnik. "BibTeXing", 1988.
Together with Appendix B of *The Manual* [104], this describes the user interface to BibTeX with useful hints for controlling its behavior.
On CTAN at: biblio/bibtex/contrib/doc/btxdoc.pdf

[136] Oren Patashnik. "Designing BibTeX styles", 1988.
A detailed description for BibTeX style designers of the postfix stack language used inside BibTeX style files. After a general description of the language, all commands and built-in functions are reviewed. Finally, BibTeX name formatting is explained in detail.
On CTAN at: biblio/bibtex/contrib/doc/btxhak.pdf

[137] John Plaice and Yannis Haralambous. "The latest developments in Ω". *TUGboat*, 17(2):181–183, 1996.
The article describes ΩTimes and ΩHelvetica, public-domain virtual Times- and Helvetica-like fonts based on real PostScript fonts, called "Glyph Containers", which will contain all necessary characters for typesetting with high TeX quality in all languages and systems using the Latin, Greek, Cyrillic, Arabic, Hebrew, and Tinagh alphabets and their derivatives. Other alphabets, such as Coptic, Armenian, and Georgian, will follow, as well as mathematical symbols, dingbats, and other character collections. Ultimately, the Ω font set will contain glyphs for the complete Unicode character set, plus some specific glyphs needed for high-quality typography.
http://www.tug.org/TUGboat/Articles/tb17-2/tb51omeg.pdf

[138] John Plaice, Yannis Haralambous, and Chris Rowley. "A multidimensional approach to typesetting". *TUGboat*. To appear.
Outline of an approach to micro-typesetting that substantially improves on that of TeX and Ω2.0. http://www.tug.org/TUGboat/Articles/...

[139] Sunil Podar. "Enhancements to the picture environment of LaTeX". Technical Report 86-17, Department of Computer Science, S.U.N.Y, 1986. Version 1.2: July 14, 1986.
This document describes some new commands for the `picture` environment of LaTeX, especially higher-level commands that enhance its graphic capabilities by providing a friendlier and more powerful user interface. This lets you create more sophisticated pictures with less effort than in basic LaTeX.

[140] Rama Porrat. "Developments in Hebrew TEX". In "Proceedings of the 7th European TEX Conference, Prague", pp. 135–147. CsTUG, Prague, 1992. ISBN 80-210-0480-0.
Discussion of available software and macro packages that support typesetting in two directions, and of associated Hebrew fonts.

[141] Bernd Raichle, Rolf Niepraschk, and Thomas Hafner. "DE-TeX-FAQ—Fragen und Antworten über TEX, LATEX und DANTE e.V.", 2003.
Frequently Asked Questions with answers about TEX and the German TEX users' Group DANTE e.V. (in German language). `http://www.dante.de/faq/de-tex-faq`

[142] Brian Reid. Scribe Document Production System User Manual. Unilogic Ltd, 1984.
The manual for the system that inspired certain aspects of LATEX.

[143] Robert M. Ritter, editor. The Oxford Style Manual. Oxford University Press, London, Oxford, New York, 2003. ISBN 0-198-60564-1.
Reference work incorporating an update to *Hart's Rules* [63], and the *Oxford Dictionary for Writers and Editors.*

[144] Tomas G. Rokicki. "A proposed standard for specials". *TUGboat*, 16(4):395–401, 1995.
A draft standard for the contents of TEX `\special` commands.
`http://www.tug.org/TUGboat/Articles/tb16-4/tb49roki.pdf`

[145] Tomas G. Rokicki. "Dvips: A DVI-to-PostScript Translator, Version 5.66a", 1997.
The user guide for dvips and its accompanying programs and packages such as afm2tfm.
On CTAN at: `dviware/dvips/dvips_man.pdf`

[146] Emmanuel Donin de Rosière. From stack removing in stack-based languages to BibTEX++. Master's thesis, ENSTBr, 2003.
A description of BibTEX++, a bibliography section creator for LATEX and a possible successor of BIBTEX. The program can compile BIBTEX `.bst` style files into Java code.
`http://www.lit.enstb.org/~keryell/eleves/ENSTBr/2002-2003/DEA/Donin_de_Rosiere`

[147] Chris Rowley. "Models and languages for formatted documents". *TUGboat*, 20(3):189–195, 1999.
Explores many ideas around the nature of document formatting and how these can be modeled and implemented. `http://www.tug.org/TUGboat/Articles/tb20-3/tb64rowl.pdf`

[148] Chris Rowley. "The LATEX legacy: 2.09 and all that". In ACM, editor, "Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing 2001, Newport, Rhode Island, United States", pp. 17–25. ACM Press, New York, NY, USA, 2001. ISBN 1-58113-383-9.
Part of a celebration for Leslie Lamport's sixtieth birthday; a very particular account of the technical history and philosophy of TEX and LATEX.

[149] Chris A. Rowley and Frank Mittelbach. "Application-independent representation of multilingual text". In Unicode Consortium, editor, "Europe, Software + the Internet: Going Global with Unicode: Tenth International

Unicode Conference, March 10–12, 1997, Mainz, Germany", The Unicode Consortium, San Jose, CA, 1997.

Explores the nature of text representation in computer files and the needs of a wide range of text-processing software. http://www.latex-project.org/papers/unicode5.pdf

[150] Richard Rubinstein. Digital Typography—An Introduction to Type and Composition for Computer System Design. Addison-Wesley, Reading, MA, USA, 1988. ISBN 0-201-17633-5. Reprinted with corrections.

This book describes a technological approach to typography. It shows how computers can be used to design, create, and position the graphical elements used to present documents on a computer.

[151] Joachim Schrod. "International LaTeX is ready to use". *TUGboat*, 11(1):87–90, 1990.

Announces some of the early standards for globalization work on LaTeX.
http://www.tug.org/TUGboat/Articles/tb11-1/tb27schrod.pdf

[152] Joachim Schrod. "An international version of *MakeIndex*". *Cahiers GUTenberg*, 10-11:81–90, 1991.

The *MakeIndex* index processor is only really usable for English texts; non-English texts, especially those using non-Latin alphabets, such as Russian, Arabic, or Chinese, prove problematic. In this case the tagging of index entries is often tedious and error prone. In particular, if markup is used within the index key, an explicit sort key must be specified. This article presents a new version of *MakeIndex*, which uses less memory so that it can be used for the creation of very large indices. It allows the automatic creation of sort keys from index keys by user-specified mappings, and supports documents in non-Latin alphabets.
http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/10-schrod.pdf

[153] Joachim Schrod. "The components of TeX". *MAPS*, 8:81–86, 1992.

TeX needs a great number of supplementary components (files and programs) whose meanings and interactions are often unknown; the structure of a complete TeX setup is explained.
http://www.ntg.nl/maps/pdf/8_18.pdf

[154] Paul Stiff. "The end of the line: A survey of unjustified typography". *Information Design Journal*, 8(2):125–152, 1996.

A good overview about the typographical problems that need to be resolved when producing high-quality unjustified copy.

[155] Anders Svensson. "Typesetting diagrams with kuvio.tex", 1996.

Manual for the kuvio system for typesetting diagrams; it uses PostScript code in \specials.
On CTAN at: macros/generic/diagrams/kuvio

[156] Ellen Swanson. Mathematics into Type. American Mathematical Society, Providence, Rhode Island, updated edition, 1999. ISBN 0-8218-1961-5. Updated by Arlene O'Sean and Antoinette Schleyer.

Originally written as a manual to standardize copyediting procedures, the second edition is also intended for use by publishers and authors as a guide in preparing mathematics copy for the printer.

[157] The *TUGboat* Team. "TeX Live CD 5 and the TeX Catalogue". *TUGboat*, 21(1):16–90, 2000.

The TeX Live CD is a ready-to-run TeX system for the most popular operating systems; it works with all major TeX-related programs and contains a complete collection of fonts, macros, and other items with support for many languages. This article describes the TeX Live CD 5 distribution with cross-references to Graham Williams' TeX catalogue.
http://www.tug.org/TUGboat/Articles/tb21-1/tb66cd.pdf
Current version: http://www.tug.org/texlive

[158] Hàn Thế Thành. "Improving TEX's typeset layout". *TUGboat*, 19(3):284–288, 1998.

This attempt to improve TEX's typeset layout is based on the adjustment of interword spacing after the paragraphs have been broken into lines. Instead of changing only the interword spacing to justify text lines, fonts on the line are also slightly expanded to minimize excessive stretching of the interword spaces. This font expansion is implemented using horizontal scaling in PDF. By using such expansion conservatively, and by employing appropriate settings for TEX's line-breaking and spacing parameters, this method can improve the appearance of TEX's typeset layout. `http://www.tug.org/TUGboat/Articles/tb19-3/tb60than.pdf`

[159] Hàn Thế Thành. "Micro-typographic extensions to the TEX typesetting system". *TUGboat*, 21(4):317–434, 2000.

Doctoral dissertation at the Faculty of Informatics, Masaryk University, Brno, Czech Republic, October 2000. `http://www.tug.org/TUGboat/Articles/tb21-4/tb69thanh.pdf`

[160] Hán Thế Thánh. "Margin kerning and font expansion with pdfTEX". *TUGboat*, 22(3):146–148, 2001.

"Margin kerning" adjusts the positions of the primary and final glyphs in a line of text to make the margins "look straight". "Font expansion" uses a slightly wider or narrower variant of a font to make interword spacing more even. These techniques are explained with the help of examples. For a detailed explanation of the concepts, see [159]. This feature was used in the preparation of this book. `http://www.tug.org/TUGboat/Articles/tb22-3/tb72thanh.pdf`

[161] Hán Thế Thánh and Sebastian Rahtz. "The pdfTEX user manual". *TUGboat*, 18(4):249–254, 1997.

UserSebastian Rahtz manual for the pdfTEX system, which extends TEX to generate PDF directly. `http://www.tug.org/TUGboat/Articles/tb18-4/tb57than.pdf`

[162] Harold Thimbleby. "'See also' indexing with makeindex". *TUGboat*, 12(2):290–290, 1991.

Describes how to produce "see also" entries with *MakeIndex* appearing after any page numbers for that entry. Also check [163].
`http://www.tug.org/TUGboat/Articles/tb12-2/tb32thim.pdf`

[163] Harold Thimbleby. "Erratum: 'See also' indexing with makeindex, *TUGboat* 12, no. 2, p. 290". *TUGboat*, 13(1):95–95, 1992.

Erratum to [162]. `http://www.tug.org/TUGboat/Articles/tb13-1/tb34thim.pdf`

[164] TUG Working Group on a TEX Directory Structure. "A directory structure for TEX files (Version 0.999)". *TUGboat*, 16(4):401–413, 1995.

Describes the commonly used standard TEX Directory Structure (TDS) for implementation-independent TEX system files. `http://www.tug.org/TUGboat/Articles/tb16-4/tb49tds.pdf`
Current version: `http://www.tug.org/tds`

[165] The Unicode Consortium. The Unicode Standard, Version 4.0. Addison-Wesley, Boston, MA, USA, 2003. ISBN 0-321-18578-1.

The reference guide of the Unicode Standard, a universal character-encoding scheme that defines a consistent way of encoding multilingual text. Unicode is the default encoding of HTML and XML. The book explains the principles of operation and contains images of the glyphs for all characters presently defined in Unicode.
Available for restricted use from: `http://www.unicode.org/versions/Unicode4.0.0`

[166] Gabriel Valiente Feruglio. "Typesetting commutative diagrams". *TUGboat*, 15(4):466–484, 1994.

Surveys the available support for typesetting commutative diagrams.
`http://www.tug.org/TUGboat/Articles/tb15-4/tb45vali.pdf`

[167] Gabriel Valiente Feruglio. "Modern Catalan typographical conventions". *TUGboat*, 16(3):329–338, 1995.

Many languages, such as German, English, and French, have a traditional typography. However, despite the existence of a well-established tradition in scientific writing in Catalan, there are not yet any standards encompassing typographical conventions in this area. This paper proposes typographical rules that reflect the spirit of ancient Catalan scientific writings while conforming to modern typographical conventions. Some of these typographical rules are incorporated in Catalan extensions to TEX and LATEX. The proposal also hopes to contribute to the development of standard rules for scientific writing in Catalan.

`http://www.tug.org/TUGboat/Articles/tb16-3/tb48vali.pdf`

[168] Michael Vulis. "VTEX enhancements to the TEX language". *TUGboat*, 11(3):429–434, 1990.

Description of the commercial VTEX system, which supports a scalable font format.

`http://www.tug.org/TUGboat/Articles/tb11-3/tb29vulis.pdf`

More recent information available from `http://www.micropress-inc.com/enfeat.htm`

[169] Graham Williams. "Graham Williams' TEX Catalogue". *TUGboat*, 21(1):17–90, 2000.

This catalogue lists more than 1500 TEX, LATEX, and related packages and tools and is linked directly to the items on CTAN.

`http://www.tug.org/TUGboat/Articles/tb21-1/tb66catal.pdf`

Latest version on CTAN at: `help/Catalogue/catalogue.html`

[170] Hugh Williamson. Methods of Book Design. Yale University Press, New Haven, London, 3rd edition, 1983.

A classic work that has become a basic tool for the practicing book designer. It deals with such matters as the preparation of copy, the selection and arrangement of type, the designer's part in book illustration and jacket design, and the economics of book production. The book also explains the materials and techniques of book production and their effect on the design of books.

[171] Peter Wilson. ledmac—A presumptuous attempt to port EDMAC and TABMAC to LATEX, 2003.

EDMAC and TABMAC are a set of plain TEX macros for typesetting critical editions in the traditional way. The ledmac package implements the facilities of these macros in LATEX—in particular, marginal line numbering and multiple series of footnotes and endnotes keyed to line numbers. As a new feature the package provides for index entries keyed to both page and line numbers. Multiple series of the familiar numbered footnotes are also available.

On CTAN at: `macros/latex/contrib/ledmac/ledmac.pdf`

[172] Reinhard Wonneberger and Frank Mittelbach. "BIBTEX reconsidered". *TUGboat*, 12(1):111–124, 1991.

A discussion of BIBTEX and several proposals for its enhancement.

`http://www.tug.org/TUGboat/Articles/tb12-1/tb31wonn.pdf`

[173] Hermann Zapf. "My collaboration with Don Knuth and my font design work". *TUGboat*, 22(1/2):26–30, 2001.

Zapf's story of collaboration with Don Knuth and some thoughts on typography.

`http://www.tug.org/TUGboat/Articles/tb22-1-2/tb70zapf.pdf`

[174] Justin Ziegler. "Technical report on math font encoding (version 2)". Technical report, LATEX3 project, 1994.

The ground work for a set of 8-bit math encodings for TEX.

On CTAN at: `info/ltx3pub/l3d007.*`

Index of Commands and Concepts

This title somewhat hides the fact that everything except the author names is in this one long index. To make it easier to use, the entries are distinguished by their "type" and this is often indicated by one of the following "type words" at the beginning of the main entry or a sub-entry:

> attribute, BIBTEX built-in function, BIBTEX command, BIBTEX entry type, BIBTEX field, BIBTEX style, boolean, counter, document class, env., env. variable, file, file extension, folio style, font, font encoding, function, key, key/option, key value, keyword, length, option, package, page style, program, rigid length, or syntax.

The absence of an explicit "type word" means that the "type" is either a LATEX "command" or simply a "concept".

Use by, or in connection with, a particular package is indicated by adding the package name (in parentheses) to an entry or sub-entry. There is one "virtual" package name, tlc, which indicates commands introduced only for illustrative purposes in this book.

A *blue italic* page number indicates that the command or concept is demonstrated in an example on that page.

When there are several page numbers listed, **bold** face indicates a page containing important information about an entry, such as a definition or basic usage.

When looking for the position of an entry in the index, you need to realize that, when they come at the start of a command or file extension, both of the characters \ and . are ignored. All symbols come before all letters and everything that starts with the `@` character will appear immediately before `A`.

Symbols

C

E

G

O

S

V

W

People

Biographies

Frank Mittelbach

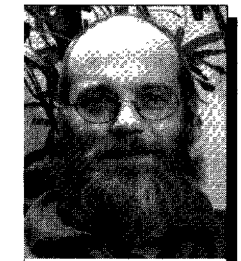
Frank Mittelbach studied mathematics and computer science at the Johannes-Gutenberg University, Mainz. In 1989 he joined EDS, Electronic Data Systems, working in a newly formed group for document processing using TeX and other tools. In his current position he is responsible for concepts and implementation for remote monitoring and management of distributed systems and networks.

before TLC2...

His interest in the automated formatting of complex documents in general, and in LaTeX in particular, goes back to his university days and has become a major interest, perhaps a vocation, and certainly it is now his "second job". He is author or co-author of many and varied LaTeX extension packages, such as AMS-LaTeX, doc, multicol, and NFSS: the New Font Selection Scheme.

At the TUG conference at Stanford University in 1989, he gave a talk about the problems with LaTeX 2.09, which led to his taking on the responsibility for the maintenance and further development of LaTeX. This effort is generally known as the LaTeX3 Project and in the capacity of technical director of this project, he has overseen the original major release of LaTeX 2_ε in 1994 and the, by now, 15 subsequent maintenance releases of this software.

His publication of many technical papers on LaTeX and on general research results in automated formatting brought him in contact with Peter Gordon from Addison-Wesley. Peter and Frank inaugurated the book series *Tools and Techniques for Computer Typesetting* (TTCT), with Frank as series editor. *The LaTeX Companion* (1994) was the first book of this series whose titles by now cover LaTeX in all its facets. Forthcoming works will expand that core to cover other typesetting and information processing tools and concepts.

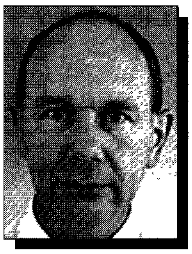


... and after

In 1990 Frank presented the paper *E-TeX: Guidelines for further TeX extensions*, which explained the most critical shortcomings of TeX and argued the need for its further development and for research into the many open questions of automated typesetting. This was the first time the topic of change or extension had been openly discussed within the TeX community and, after getting some early opposition, it helped to spawn several important projects, such as eTeX, Omega, and NTS. He is now interested in bringing together the fruits of these TeX extension developments, e.g., the Omega and eTeX projects, to get a stable, well-maintained, and widely available successor of TeX on which a future LaTeX3 can be based.

Frank lives with his wife, Christel, and their three sons, Arno (age 19) and the twins Burkhard and Holger (age 6), in Mainz, Germany.

Michel Goossens



After finishing his Ph.D. in high energy physics Michel Goossens joined CERN, the European Laboratory for Particle Physics in Geneva (Switzerland) at the beginning of 1979, where he worked for a few years as a research physicist, and then moved on to software support in the Informatics Technologies Division.

Over the years he has worked with several typesetting systems: LaTeX, of course, but also, more recently, HTML/SGML/XML. As a large international scientific laboratory, a large fraction of the thousands of physicists and engineers working at CERN use LaTeX for publishing their papers or for writing their documentation. Therefore, since the late 80s Michel has been involved in developing and supporting tools related to TeX and, especially, LaTeX.

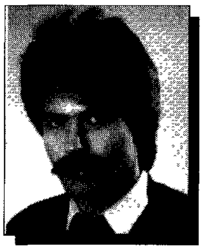
A milestone in his LaTeX life was a meeting with Frank and Chris at CERN at the end of 1992, where they gave a talk on LaTeX3. After their seminar Michel showed them the "Local TeX Guide" that he and Alexander Samarin had written and proposed to extend the material and turn it into a book. This was the birth of the first edition of *The LaTeX Companion*, which was published at the beginning of 1994. Using his experience in graphics and web presentation, he also co-authored *The LaTeX Graphics Companion* (1997) and *The LaTeX Web Companion* (1999), both of which appeared in the TTCT series.

Michel has occupied various positions in the TeX world. He was president of GUTenberg, the French-speaking TeX users Group (1995–2000), as well as president of TUG, the TeX Users Group (1995–1997).

For the past three years he has acted as the CERN Focal Point for the EU-funded TIPS (Tools for Innovative Publishing in Science) project. Within the framework of that project he was responsible for studying how XML tools can be optimally integrated into a framework for efficiently handling electronic information, especially for scientific documents. In particular, he looked at the complementary roles played by LaTeX and MathML for mathematics, SVG for graphics, PDF for typographic quality output, and XHTML or DocBook for structural integration in the Web environment.

He lives in the Geneva area and enjoys reading, watching a good film, walking along the lake or in the beautiful countryside, and visiting museums.

Johannes Braams



Johannes Braams studied electronic engineering at the Technical University in Enschede, the Netherlands. His master's thesis was on video encoding, based on a model of the human visual system. He first met LaTeX at the *dr. Neher Laboratories* of the Dutch PTT in 1984. He was a founding board member of the Dutch speaking TeX User Group (NTG) in 1988 and participated in developing support for typesetting Dutch documents.

He started work on the babel system following the Karlsruhe EuroTeX conference in 1989 and has been a member of the LaTeX3 project since the EuroTeX conference at Cork in 1990. In addition to babel, Johannes is the current maintainer of a number of LaTeX extension packages, such as the ntgclass family of document classes, the supertabular package, and the changebar package.

Johannes is still working for the Dutch PTT, nowadays known as KPN, primarily as a project manager for IT related projects. He lives with his wife, Marion, and two sons, Tycho (age 11) and Stephan (age 9), in Zoetermeer.

David Carlisle

David Carlisle studied mathematics at the University of Manchester and then worked as a researcher in the Mathematics and Computer Science departments at Cambridge and Manchester, where he started using LaTeX in 1987. He joined the LaTeX3 team in 1992, just prior to the start of development work on LaTeX 2_ε.

For the last six years he has worked at NAG Ltd. in Oxford, UK, primarily on projects connected to the development of XML-based languages for the representation of mathematical expressions and documents. He is an editor of the OpenMath specification and was an invited expert on the W3C

Math Working Group responsible for MathML, becoming an editor of the MathML 2 Recommendation. Currently he is an editor of a proposed update to ISO/IEC TR 9573, the "ISO character entities". This allows a wide range of characters to be entered into XML and SGML documents using only ASCII characters, with syntax such as γ to denote γ.

David has also taken an interest in the XSLT language and is a major contributor to the xsl-list discussion group for that language. He has reviewed or acted as technical editor on several XSLT-related books. He lives in Oxfordshire with his wife, Joanna, and their son, Matthew (4 months).

Chris Rowley

When not indulging his addiction to travel, Chris lives in London with his wine cellar, his ceramic collection, and his memories. These last include some now rather hazy ones of the 60s, when he was addicted to mathematics but also dipped his mind into computing, both the theory of programming (pretty wild stuff back then) and number crunching (nice streamers from the paper tape).

at his favourite task

It was not until the early 80s that he discovered, on a newly occupied desk, a TV-like object that was connected to a computer and could help him do creative and useful things, such as producing a single page of beautiful typeset mathematics. That was not done using TeX—so it took two days to complete that single page; but it made him realize what was possible and set him thinking about a better way to achieve it. He is very grateful that he then very soon stumbled across TeX and, not long after, LaTeX; the latter being especially providential, as his colleagues included six mathematical typists who needed something that would work for them too. A few years on he heard about a guy called Mittlebach-and-Schöpf (sic) in Mainz and the rest is ... *to be continued.*

Fifteen years later and Chris Rowley is now a senior member of the Faculty of Mathematics and Computing at the Open University, UK. He has been a manager and active member of the LaTeX3 Project Team since its beginning, when he foolishly believed that it would all be done in two years or so. He has been on too many boards and committees, one of the most pleasant being the editorial board for *Tools and Techniques for Computer Typesetting*, and he has graced various offices in the TeX world, including Chair of UKTUG and a vice-presidency of TUG.

As the largest international player in industrialized mass education for home- and workplace-based university-level customers, the Open University has become a major multi-media publishing corporation with, despite commercial competition, an under-resourced, LaTeX-based production system for its mathematical output. As a mathematician who already understood a fair bit about the production of mathematical texts, Chris was well placed to play a vital rôle in the political, administrative, and technical aspects of establishing this system in the mid-80s.

He is now actively engaged on research into the automation of all aspects of document processing, especially multi-lingual typography for multi-use documents. By contrast, over the decades he has also done his share of practical work on LaTeX-based systems in production environments and acted as consultant on the digitization of mathematical texts to a number of standards bodies, companies, and organizations.

These activities have led Chris to the conviction that TeX has but two important long-term future uses: one is as a vernacular within less formal electronic communications between mathematicians, whilst the other is as a treasure trove of wonderful algorithms, especially for mathematical typesetting. He believes, moreover, that extending the monolithic design and intricate models of the TeX software system will not lead to powerful and flexible typesetting software for the 21st Century, ... but it's more fun than doing crosswords.

Christine Detig & Joachim Schrod

In 1982, Christine Detig met TeX on reel-tape during her computer science studies, resulting in her becoming a founding member of DANTE, the German TeX Users Group. Her early software experiences were gained around the TeX workbench, resulting in the formation of a small business in the provision of TeX distributions. Spreading TeX knowledge as part of her job as a research assistant at TU Darmstadt resulted in a book for TeX beginners: *Der LaTeX Wegweiser*. Meanwhile, visiting lots of international conferences has led to many friendships with the eclectic crowd of TeXies. Meet her there for a nice chat about the Future of TeX!

Joachim Schrod also started to use TeX in 1982 and he is another founding member of DANTE. He wrote and supported the international version of LaTeX until LaTeX 2_ε came along. He has been involved in lots of TeX activities, most of them too long ago to be remembered, but among the more enduring are the creation of CTAN and the TeX Directory Structure. Today he is the CEO of a consulting company, where he strives to translate between business and technical people.

Christine & Joachim live in Rödermark, Germany.

Production Notes

This book was typeset using the LaTeX document processing system, which it describes, together with substantial help from some of the extension packages it covers, and considerable extra ad hoc LaTeX programming effort.

The text body font used is Lucida Bright (Bigelow/Holmes) at 8.8pt/12pt. The other major font is the mono-spaced European Modern Typewriter (Y&Y) 10.06pt/12pt. This particular combination was chosen to get a reasonable amount of material on each page and to optically balance the appearance of the "typewriter font" so that it was distinguishable but without too big a contrast. *Body fonts*

The text in the examples mostly uses Adobe's Times Roman with Helvetica for sans serif. For the mathematical material in the examples we have used the by now classic Computer Modern math fonts, so the symbols will appear familiar to the majority of mathematics users. Of course, examples intended to demonstrate the use of other fonts are exceptions. *Example fonts*

The book was typeset with the base LaTeX release dated 2003/12/01. The pdfTeX program was used as the underlying engine, but it was not set to produce PDF output: we were more interested in its ability to produce "hanging punctuation", and this typographical icing (package pdfcprot) was used for the main galley text (see [159, 160] for a description of how this is implemented). For comparison look at pages 941–943, as these are set without hanging punctuation (and in smaller type). *Hanging punctuation*

The production of this book required custom class and package files. It also needed a complex "make" process using a collection of "shell scripts" controlled by a "Makefile". One of the major tasks these accomplished was to ensure that the typeset output of each and every example really is produced by the accompanying example input. *The production cycle*

This "make" process worked as follows:

Generating examples

- When first processing a chapter, LaTeX generated a source document file for each example. These are the "example files" you will find on the CD-ROM.
- The make process then ran each of these "example files" through LaTeX (also calling BibTeX or whatever else was needed) as often as was necessary to produce the final form of the typeset output.

  Finally it used dvips to produce either one or two EPS files containing the "typeset example".
- The next time LaTeX was run on that chapter, each of these EPS output files was automatically placed in its position in the book, next to (or near) the example input. The process was not complete even then because the horizontal positioning of some elements, in particular the examples, depends on whether they are on a verso or recto (the technique from Example A-3-9 on page 876 was used in this case). Thus, at least one or two additional runs were needed before all the cross-references were correctly resolved and LaTeX finally found the right way to place the examples correctly into the margins.

Manual labor

That was about as far as automation of the process could take us. Because of the many large examples that could neither be broken nor treated as floating material, getting good page breaks turned out to be a major challenge. For this and other reasons, getting to the final layout of the book was fairly labor intensive and even required minor rewriting (on maybe 10% of the pages) in order to avoid bad line breaks or page breaks (e.g., paragraphs ending with a single word line or a distracting hyphenation at a page break). Spreads were allowed to run one line long or one line short if necessary and in several cases the layout and contents of the examples were manually adjusted to allow decent page breaks.

Some statistics

Here are a few approximate statistics from this page layout process: 45 long spreads, 25 short spreads, 230 forced page breaks, 400 adjustments to the vertical spacing, 100 other manual adjustments (other than rewriting).

The index

The "Commands and Concepts" index was produced by printing a version of the book with line numbers and giving that to the indexer, who produced "conceptual index entries" that were then added to the source files for the book. This was a major testament to the quality of the lineno package, as it worked "straight out of the box". For the index processing *MakeIndex* was used as xindy was not then available. However, due to the complexity of the index (the colored page numbers, etc.) it was necessary to use pre- and post-processing by scripts to produce the final form of the index file. This was then typeset using an enhanced version of the `multicol` package to add the continuation lines—something that perhaps one day can be turned into a proper package.