

References for T_EX and Friends

Michael Wiedmann

V0.3.6 2004-06-21

Revision History

Revision 0.3.6 2004-06-21

PDF file optimized/linearized and some finetuning; corrected typos and added missing hyperlinks in chapter xindy; added a note in chapter L^AT_EX with the copyright and permission notice of the original file latex2.texi.

Revision 0.2.0 2003-01-11

License changed to GNU Free Documentation License <<http://www.gnu.org/licenses/fdl.html>>.

Revision 0.0.1 2002-06-21

This version was adapted from the edition 1.6 of the L^AT_EX 2_ε documentation, converted to DocBook XML using texi2db <<http://www.tldp.org/downloads/>>, and further edited manually.

Copyright

Copyright © ? Stephen Gilmore

Copyright © ? Torsten Martinsen

Copyright © 1988, 1994 Free Software Foundation, Inc.

Copyright © 1994 – 1996 Torsten Martinsen

Copyright © 2002, 2003 Peter Karp, Michael Wiedmann

Copyright © 2004 Michael Wiedmann

License

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

Preface

L^AT_EX 2_ε is a document preparation system implemented as a macro package for Donald E. Knuth's T_EX typesetting program.

L^AT_EX was originally conceived by Leslie Lamport.

This updated L^AT_EX reference is by no means complete, but it's a first step towards a more complete L^AT_EX reference! We welcome if we'll get pointed to missing commands or even better when you make additions and further improvements and send them to the authors.

TUG (T_EX User Group) <<http://www.tug.org>> and DANTE (German T_EX User Group) <<http://www.dante.de>> both have excellent FAQ's which answer many questions or give great pointers to additional resources both on the internet or in books. Most if not all missing packages can also be found on the CTAN servers.

This reference is not meant to replace the package documentation or the standard literature like the L^AT_EX handbook and the L^AT_EX companion. Look there for detailed descriptions of the commands and even mostly more options and examples.

The authors intend to extend this documentation to other T_EX related packages (e.g. ConTeXt, MetaPost, Metafont, etc.). Contributions are very welcome!

Credits

The following people have contributed substantial parts to this documentation project or helped in some other way (listed alphabetically):

- Karl Berry
- Wolfgang Borgert
- Ramon Casellas
- James Devenish
- Hubert Gaesslein
- Peter Karp
- Rolf Niepraschk
- Simon Pepping
- Bob Stayton
- Herbert Voss

Table of Contents

1.	TeX	9
1.1.	PlainTeX	9
1.1.1.	Overview	9
1.2.	L ^A T _E X	9
1.2.1.	Overview of L ^A T _E X and Local Guide	9
1.2.2.	Commands	9
1.2.3.	Parameters	45
1.3.	L ^A T _E X Packages	45
1.3.1.	amsmath	45
1.3.2.	dcolumn	46
1.3.3.	fontenc	46
1.3.4.	german	46
1.3.5.	graphics	47
1.3.6.	hyperref	47
1.3.7.	inputenc	50
1.3.8.	KOMA-Script	51
1.3.9.	mathpazo	59
1.3.10.	varioref	60
1.3.11.	Several Small Packages	60
1.4.	Generic Packages	60
1.4.1.	P _T CT _E X (tbd.)	60
1.4.2.	PSTricks	61
1.5.	fontinst (tbd.)	71
1.6.	ConTeXt (tbd.)	71
1.6.1.	Overview	71
1.7.	Texinfo (tbd.)	71
2.	pdfTeX	72
2.1.	Options	72
2.1.1.	--file-line-error-style	72
2.1.2.	--fmt format	72
2.1.3.	--help	72
2.1.4.	--ini	72
2.1.5.	--interaction mode	72
2.1.6.	--ipc	72
2.1.7.	--ipc-start	72
2.1.8.	--jobname name	72
2.1.9.	--kpathsea-debug bitmask	73
2.1.10.	--maketex fmt	73
2.1.11.	--no-maketex fmt	73
2.1.12.	--output-comment string	73
2.1.13.	--parse-first-line	73
2.1.14.	--progname name	73
2.1.15.	--recorder	73
2.1.16.	--shell-escape	73
2.1.17.	--translate-file tcxname	73
2.1.18.	--version	73
2.2.	Environment	73
3.	Metafont (tbd.)	75
4.	MetaPost	76
4.1.	Options	76
4.1.1.	--file-line-error-style	76
4.1.2.	--help	76
4.1.3.	--ini	76
4.1.4.	--interaction mode	76
4.1.5.	--jobname name	76
4.1.6.	--kpathsea-debug bitmask	76
4.1.7.	--mem mem	76

4.1.8.	<code>--parse-first-line</code>	77
4.1.9.	<code>--progname name</code>	77
4.1.10.	<code>--recorder</code>	77
4.1.11.	<code>-T</code>	77
4.1.12.	<code>--translate-file tcxname</code>	77
4.1.13.	<code>--troff</code>	77
4.1.14.	<code>--version</code>	77
4.2.	Environment	77
5.	BibTeX	79
5.1.	Parameters	79
5.2.	Command Qualifiers	79
5.3.	bib files	79
5.3.1.	<code>@STRING</code> command	79
5.3.2.	Entry Format	80
5.3.3.	Entry Types	80
5.3.4.	Field Text	84
5.3.5.	Field Types	86
5.4.	bst files	89
5.4.1.	ABBRV.BST	89
5.4.2.	ALPHA.BST	89
5.4.3.	PLAIN.BST	89
5.4.4.	SAMPLE.BST	89
5.4.5.	UNSRT.BST	90
6.	MakeIndex	91
6.1.	Options	91
6.1.1.	<code>-c</code>	91
6.1.2.	<code>-g</code>	91
6.1.3.	<code>-i</code>	91
6.1.4.	<code>-l</code>	91
6.1.5.	<code>-q</code>	91
6.1.6.	<code>-r</code>	92
6.1.7.	<code>-s sty</code>	92
6.1.8.	<code>-o ind</code>	92
6.1.9.	<code>-t log</code>	92
6.1.10.	<code>-p no</code>	92
6.1.11.	<code>-L</code>	92
6.1.12.	<code>-T</code>	92
6.2.	Style File	93
6.2.1.	Input Style Specifiers	93
6.2.2.	Output Style Specifiers	94
6.3.	Example	98
6.4.	Ordering	98
6.5.	Special Effects	98
7.	xindy	101
7.1.	Command List	101
7.1.1.	Processing Commands	101
7.1.2.	Markup Commands	107
7.1.3.	Raw Index Interface	112
7.2.	Invoking xindy	113
7.2.1.	Command Line Options	113
7.2.2.	Search Path	114
A.	Appendices	115
A.1.	Bugs / Known Issues / Missing Features	115
A.2.	About this Document	115
A.3.	Release News	115
A.4.	GNU Free Documentation License	117

Index

124

1. T_EX

1.1. PlainT_EX

1.1.1. Overview

For a really excellent reference of all T_EX primitive control sequences see David Bausum's *T_EX Primitive Control Sequences* <<http://www.tug.org/utilities/plain/cseq.html>>.

1.2. L^AT_EX

NOTE

The original file `latex2e.texi` contains the following copyright and permission notice:

"Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies."

"Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one."

1.2.1. Overview of L^AT_EX and Local Guide

The L^AT_EX command typesets a file of text using the T_EX program and the L^AT_EX Macro package for T_EX. To be more specific, it processes an input file containing the text of a document with interspersed commands that describe how the text should be formatted. It produces at least three files as output:

1. A *Device Independent*, or `.dvi` file. This contains commands that can be translated into commands for a variety of output devices. You can view the output of L^AT_EX by using a program such as `xdvi`, which actually uses the `.dvi` file.
2. A *transcript* or `.log` file that contains summary information and diagnostic messages for any errors discovered in the input file.
3. An *auxiliary* or `.aux` file. This is used by L^AT_EX itself, for things such as sectioning.

For a description of what goes on inside T_EX, you should consult *The TeXbook* by Donald E. Knuth, ISBN 0-201-13448-9, published jointly by the American Mathematical Society and Addison-Wesley Publishing Company.

For a description of L^AT_EX, you should consult:

L^AT_EX: A Document Preparation System, by Leslie Lamport, Addison-Wesley Publishing Company, 2nd edition, 1994.

The L^AT_EX Companion, by Michel Goossens, Frank Mittelbach, and Alexander Samarin, Addison-Wesley, 1994.

1.2.2. Commands

A L^AT_EX command begins with the command name, which consists of a `\` followed by either (a) a string of letters or (b) a single non-letter. Arguments contained in square brackets, `[]`, are optional while arguments contained in braces, `{}`, are required.

NOTE: L^AT_EX is case sensitive. Enter all commands in lower case unless explicitly directed to do otherwise.

1.2.2.1. Counters

Everything L^AT_EX numbers for you has a counter associated with it. The name of the counter is the same as the name of the environment or command that produces the number, except with no `\` (`enumi` - `enumiv` are used for the nested enumerate environment). Below is a list of the counters used in L^AT_EX's standard document classes to control numbering.

<code>part</code>	<code>paragraph</code>	<code>figure</code>	<code>enumi</code>
<code>chapter</code>	<code>subparagraph</code>	<code>table</code>	<code>enumii</code>
<code>section</code>	<code>page</code>	<code>footnote</code>	<code>enumiii</code>
<code>subsection</code>	<code>equation</code>	<code>mpfootnote</code>	<code>enumiv</code>
<code>subsubsection</code>			

`\addtocounter` The `\addtocounter` command increments the counter by the amount specified by the `value` argument. The `value` argument can be negative.

`\alph{counter}` This command causes the value of the counter to be printed in alphabetic characters. The `\alph` command uses lower case alphabetic characters, i.e., `a`, `b`, `c`... while the `\Alph` command uses upper case alphabetic characters, i.e., `A`, `B`, `C`...

`\arabic{counter}` The `\arabic` command causes the value of the counter to be printed in Arabic numbers, i.e., `3`.

`\fnsymbol{counter}` The `\fnsymbol` command causes the value of the counter to be printed in a specific sequence of nine symbols that can be used for numbering footnotes.

NB. counter must have a value between 1 and 9 inclusive.

`\newcounter{foo}[counter]` The `\newcounter` command defines a new counter named `foo`. The counter is initialized to zero.

The optional argument `[counter]` causes the counter `foo` to be reset whenever the counter named in the optional argument is incremented.

`\refstepcounter{counter}` The `\refstepcounter` command works like `\stepcounter` See `\stepcounter`, except it also defines the current `\ref` value to be the result of `\thecounter`.

`\roman{counter}` This command causes the value of the counter to be printed in Roman numerals. The `\roman` command uses lower case Roman numerals, i.e., `i`, `ii`, `iii`..., while the `\Roman` command uses upper case Roman numerals, i.e., `I`, `II`, `III`...

`\stepcounter{counter}` The `\stepcounter` command adds one to the counter and resets all subsidiary counters.

`\setcounter{counter}{value}` The `\setcounter` command sets the value of the counter to that specified by the `value` argument.

`\usecounter{counter}` The `\usecounter` command is used in the second argument of the `list` environment to allow the counter specified to be used to number the list items.

`\value{counter}` The `\value` command produces the value of the `counter` named in the mandatory argument. It can be used where L^AT_EX expects an integer or number, such as the second argument of a `\setcounter` or `\addtocounter` command, or in:

```
\hspace{\value{foo}\parindent}
```

It is useful for doing arithmetic with counters.

1.2.2.2. Cross References

One reason for numbering things like figures and equations is to refer the reader to them, as in *See Figure 3 for more details*. A smarter version of the cross reference commands is available in the package `varioref`.

`\label{key}` A `\label` command appearing in ordinary text assigns to the `key` the number of the current sectional unit; one appearing inside a numbered environment assigns that number to the `key`.

A `key` can consist of any sequence of letters, digits, or punctuation characters. Upper and lowercase letters are different.

To avoid accidentally creating two labels with the same name, it is common to use labels consisting of a prefix and a suffix separated by a colon. The prefixes conventionally used are

- `cha` for chapters
- `sec` for lower-level sectioning commands
- `fig` for figures
- `tab` for tables
- `eq` for equations

Thus, a label for a figure would look like `fig:bandersnatch`.

`\pageref{key}` The `\pageref` command produces the page number of the place in the text where the corresponding `\label` command appears. ie. where `\label{key}` appears. For a smart version see `\vpageref`.

`\ref{key}` The `\ref` command produces the number of the sectional unit, equation number, ... of the corresponding `\label` command. For a smart version see `\vref`.

1.2.2.3. Definitions

`\newcommand`

```
\newcommand{cmd}[args]{definition}
\newcommand{cmd}[args][default]{definition}
\renewcommand{cmd}[args]{definition}
\renewcommand{cmd}[args][default]{definition}
```

These commands define (or redefine) a command.

`\newenvironment`

```
\newenvironment{nam}[args]{begdef}{enddef}
\newenvironment{nam}[args][default]{begdef}{enddef}
\renewenvironment{nam}[args]{begdef}{enddef}
```

These commands define or redefine an environment.

cmd	A command name beginning with a <code>\</code> . For <code>\newcommand</code> it must not be already defined and must not begin with <code>\end</code> ; for <code>\renewcommand</code> it must already be defined.
args	An integer from 1 to 9 denoting the number of arguments of the command being defined. The default is for the command to have no arguments.
def	If this optional parameter is present, it means that the command's first argument is optional. The default value of the optional argument is <code>def</code> .
definition	The text to be substituted for every occurrence of <code>cmd</code> ; a parameter of the form <code>#n</code> in <code>cmd</code> is replaced by the text of the <code>n</code> th argument when this substitution takes place.

Table 1.1.: Options for `\newcommand`

nam	The name of the environment. For <code>\newenvironment</code> there must be no currently defined environment by that name, and the command <code>\nam</code> must be undefined. For <code>\renewenvironment</code> the environment must already be defined.
args	An integer from 1 to 9 denoting the number of arguments of the newly-defined environment. The default is no arguments. Arguments can only be used in the <code>{begdef}</code> part.
default	If this is specified, the first argument is optional, and <code>default</code> gives the default value for that argument.
begdef	The text substituted for every occurrence of <code>\begin{nam}</code> ; a parameter of the form <code>#n</code> in <code>cmd</code> is replaced by the text of the <code>n</code> th argument when this substitution takes place.
enddef	The text substituted for every occurrence of <code>\end{nam}</code> . It may not contain any argument parameters.

Table 1.2.: Options for `\newenvironment`

`\newtheorem`

`\newtheorem{env_name}{caption}[within]`

`\newtheorem{env_name}[numbered_like]{caption}`

This command defines a theorem-like environment.

env_name	The name of the environment to be defined. A string of letters. It must not be the name of an existing environment or counter.
caption	The text printed at the beginning of the environment, right before the number. This may simply say <i>Theorem</i> , for example.
within	The name of an already defined counter, usually of a sectional unit. Provides a means of resetting the new theorem counter within the sectional unit.
numbered_like	The name of an already defined theorem-like environment.

Table 1.3.: Options for `\newtheorem`

The `\newtheorem` command may have at most one optional argument.

`\newfont \newfont{cmd}{font_name}`

Defines the command name `cmd`, which must not be currently defined, to be a declaration that selects the font named `font_name` to be the current font.

1.2.2.4. Document Classes

Valid L^AT_EX document classes include:

- article
- report
- letter
- book
- slides

Other document classes are often available. See Overview, for details. They are selected with the following command:

`\documentclass [options] {class}`

All the standard classes (except slides) accept the following options for selecting the typeface size (10pt is default):

10pt, 11pt, 12pt

All classes accept these options for selecting the paper size (default is letter):

a4paper, a5paper, b5paper, letterpaper, legalpaper, executivepaper

Miscellaneous options:

- landscape — selects landscape format. Default is portrait.
- titlepage, notitlepage — selects if there should be a separate title page.
- leqno — equation number on left side of equations. Default is right side.
- fleqn — displayed formulas flush left. Default is centered.
- openbib — use *open* bibliography format.
- draft, final — mark/do not mark overfull boxes with a rule. Default is final.

These options are not available with the slides class:

- oneside, twoside — selects one- or twosided layout. Default is oneside, except for the book class.
- openright, openany — determines if a chapter should start on a right-hand page. Default is openright for book.
- onecolumn, twocolumn — one or two columns. Defaults to one column.

The slides class offers the option `clock` for printing the time at the bottom of each note.

If you specify more than one option, they must be separated by a comma.

Additional packages are loaded by a

`\usepackage [options] {pkg}`

command. If you specify more than one package, they must be separated by a comma.

Any options given in the `\documentclass` command that are unknown by the selected document class are passed on to the packages loaded with `\usepackage`.

1.2.2.5. Layout

Miscellaneous commands for controlling the general layout of the page.

`\flushbottom` The `\flushbottom` declaration makes all text pages the same height, adding extra vertical space when necessary to fill out the page.

This is the standard if twocolumn mode is selected.

`\onecolumn` The `\onecolumn` declaration starts a new page and produces single-column output.

`\raggedbottom` The `\raggedbottom` declaration makes all pages the height of the text on that page. No extra vertical space is added.

`\twocolumn` `\twocolumn [text]`

The `\twocolumn` declaration starts a new page and produces two-column output. If the optional `text` argument is present, it is typeset in one-column mode.

1.2.2.6. Environments

L^AT_EX provides a number of different paragraph-making environments. Each environment begins and ends in the same manner.

```
\begin{environment-name}
.
.
.
\end{environment-name}
```

array

```
\begin{array}{col1col2...coln}
column 1 entry & column 2 entry ... & column n entry \\
.
.
.
\end{array}
```

Math arrays are produced with the `array` environment. It has a single mandatory argument describing the number of columns and the alignment within them. Each column, `coln`, is specified by a single letter that tells how items in that row should be formatted.

- `c` — for centered
- `l` — for flush left
- `r` — for flush right

Column entries must be separated by an `&`. Column entries may include other L^AT_EX commands. Each row of the array must be terminated with the string `\\`.

Note that the `array` environment can only be used in math mode, so normally it is used inside an `equation` environment.

center

```
\begin{center}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{center}
```

The `center` environment allows you to create a paragraph consisting of lines that are centered within the left and right margins on the current page. Each line must be terminated with the string `\\`.

`\centering` This declaration corresponds to the `center` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`. The text of a figure or table can be centered on the page by putting a `\centering` command at the beginning of the figure or table environment.

Unlike the `center` environment, the `\centering` command does not start a new paragraph; it simply changes how L^AT_EX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like `quote`) that ends the paragraph unit.

description

```
\begin{description}
\item [label] First item
\item [label] Second item
.
.
.
\end{description}
```

The `description` environment is used to make labelled lists. The `label` is bold face and flushed right.

`enumerate`

```
\begin{enumerate}
\item First item
\item Second item
.
.
.
\end{enumerate}
```

The `enumerate` environment produces a numbered list. Enumerations can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments.

Each item of an enumerated list begins with an `\item` command. There must be at least one `\item` command within the environment.

The `enumerate` environment uses the `enumi` through `enumiv` counters (see Counters). The type of numbering can be changed by redefining `\theenumi` etc.

`eqnarray`

```
\begin{eqnarray}
math formula 1 \\
math formula 2 \\
.
.
.
\end{eqnarray}
```

The `eqnarray` environment is used to display a sequence of equations or inequalities. It is very much like a three-column `array` environment, with consecutive rows separated by `\\` and consecutive items within a row separated by an `&`.

An equation number is placed on every line unless that line has a `\nonumber` command.

The command `\lefteqn` is used for splitting long formulas across lines. It typesets its argument in display style flush left in a box of zero width.

`equation`

```
\begin{equation}
math formula
\end{equation}
```

The `equation` environment centers your equation on the page and places the equation number in the right margin.

`figure`

```
\begin{figure}[placement]

body of the figure

\caption{figure title}
\end{figure}
```

Figures are objects that are not part of the normal text, and are usually *float*ed to a convenient place, like the top of a page. Figures will not be split between two pages.

The optional argument [**placement**] determines where L^AT_EX will try to place your figure. There are four places where L^AT_EX can possibly put a float:

1. **h** (Here) - at the position in the text where the figure environment appears.
2. **t** (Top) - at the top of a text page.
3. **b** (Bottom) - at the bottom of a text page.
4. **p** (Page of floats) - on a separate float page, which is a page containing no text, only floats.

The standard report and article classes use the default placement **tbp**.

The body of the figure is made up of whatever text, L^AT_EX commands, etc. you wish. The `\caption` command allows you to title your figure.

`flushleft`

```
\begin{flushleft}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{flushleft}
```

The `flushleft` environment allows you to create a paragraph consisting of lines that are flushed left, to the left-hand margin. Each line must be terminated with the string `\\`.

`\raggedright` This declaration corresponds to the `flushleft` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushleft` environment, the `\raggedright` command does not start a new paragraph; it simply changes how L^AT_EX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like `quote`) that ends the paragraph unit.

`flushright`

```
\begin{flushright}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{flushright}
```

The `flushright` environment allows you to create a paragraph consisting of lines that are flushed right, to the right-hand margin. Each line must be terminated with the string `\\`.

`\raggedleft` This declaration corresponds to the `flushright` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushright` environment, the `\raggedleft` command does not start a new paragraph; it simply changes how L^AT_EX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like `quote`) that ends the paragraph unit.

itemize

```

\begin{itemize}
\item First item
\item Second item
.
.
.
\end{itemize}

```

The `itemize` environment produces a *bulleted* list. Itemizations can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments.

Each item of an `itemized` list begins with an `\item` command. There must be at least one `\item` command within the environment.

The `itemize` environment uses the `itemi` through `itemiv` counters (see `Counters`). The type of numbering can be changed by redefining `\theitemi` etc.

letter This environment is used for creating letters. See `Letters`.

list The `list` environment is a generic environment which is used for defining many of the more specific environments. It is seldom used in documents, but often in macros.

```

\begin{list}{label}{spacing}
\item First item
\item Second item
.
.
.
\end{list}

```

The `{label}` argument specifies how items should be labelled. This argument is a piece of text that is inserted in a box to form the label. This argument can and usually does contain other L^AT_EX commands.

The `{spacing}` argument contains commands to change the spacing parameters for the list. This argument will most often be null, i.e., `{}`. This will select all default spacing which should suffice for most cases.

minipage

```

\begin{minipage}[position][height][inner-pos]{width}
text
\end{minipage}

```

The `minipage` environment is similar to a `\parbox` command. It takes the same optional `position` argument and mandatory `width` argument. You may use other paragraph-making environments inside a `minipage`.

Footnotes in a `minipage` environment are handled in a way that is particularly useful for putting footnotes in figures or tables. A `\footnote` or `\footnotetext` command puts the footnote at the bottom of the `minipage` instead of at the bottom of the page, and it uses the `mpfootnote` counter instead of the ordinary `footnote` counter. See `Counters`.

NOTE: Don't put one `minipage` inside another if you are using footnotes; they may wind up at the bottom of the wrong `minipage`.

picture

```

\begin{picture}(width,height)(x offset,y offset)
.
.

```

```

picture commands
.
.
\end{picture}

```

The `picture` environment allows you to create just about any kind of picture you want containing text, lines, arrows and circles. You tell L^AT_EX where to put things in the picture by specifying their coordinates. A coordinate is a number that may have a decimal point and a minus sign — a number like 5, 2.3 or -3.1416. A coordinate specifies a length in multiples of the unit length `\unitlength`, so if `\unitlength` has been set to 1cm, then the coordinate 2.54 specifies a length of 2.54 centimetres. You can change the value of `\unitlength` anywhere you want, using the `\setlength` command, but strange things will happen if you try changing it inside the picture environment.

A position is a pair of coordinates, such as (2.4,-5), specifying the point with x-coordinate 2.4 and y-coordinate -5. Coordinates are specified in the usual way with respect to an origin, which is normally at the lower-left corner of the picture. Note that when a position appears as an argument, it is not enclosed in braces; the parentheses serve to delimit the argument.

The `picture` environment has one mandatory argument, which is a **position**. It specifies the size of the picture. The environment produces a rectangular box with width and height determined by this argument's x- and y-coordinates.

The `picture` environment also has an optional **position** argument, following the **size** argument, that can change the origin. (Unlike ordinary optional arguments, this argument is not contained in square brackets.) The optional argument gives the coordinates of the point at the lower-left corner of the picture (thereby determining the origin). For example, if `\unitlength` has been set to 1mm, the command

```
\begin{picture}(100,200)(10,20)
```

produces a picture of width 100 millimetres and height 200 millimetres, whose lower-left corner is the point (10,20) and whose upper-right corner is therefore the point (110,220). When you first draw a picture, you will omit the optional argument, leaving the origin at the lower-left corner. If you then want to modify your picture by shifting everything, you just add the appropriate optional argument.

The environment's mandatory argument determines the nominal size of the picture. This need bear no relation to how large the picture really is; L^AT_EX will happily allow you to put things outside the picture, or even off the page. The picture's nominal size is used by L^AT_EX in determining how much room to leave for it.

Everything that appears in a picture is drawn by the `\put` command. The command

```
\put (11.3,-.3){...}
```

puts the object specified by `...` in the picture, with its reference point at coordinates (11.3,-.3). The reference points for various objects will be described below.

The `\put` command creates an **LR box**. You can put anything in the text argument of the `\put` command that you'd put into the argument of an `\mbox` and related commands. When you do this, the reference point will be the lower left corner of the box.

Picture commands:

`\circle{diameter}`, `\circle*{diameter}` The `\circle` command produces a circle with a diameter as close to the specified one as possible. If the `*`-form of the command is used, L^AT_EX draws a solid circle.

Note that only circles up to 40 pt can be drawn.

`\dashbox` Draws a box with a dashed line.

```
\dashbox{dash_length}(width,height){...}
```

The `\dashbox` has an extra argument which specifies the width of each dash. A dashed box looks best when the `width` and `height` are multiples of the `dash_length`.

`\frame{...}` The `\frame` command puts a rectangular frame around the object specified in the argument. The reference point is the bottom left corner of the frame. No extra space is put between the frame and the object.

`\framebox(width,height)[position]{...}` The `\framebox` command is exactly the same as the `\makebox` command, except that it puts a frame around the outside of the box that it creates.

The `\framebox` command produces a rule of thickness `\fboxrule`, and leaves a space `\fboxsep` between the rule and the contents of the box.

`\line(x slope,y slope){length}` The `\line` command draws a line of the specified `length` and `slope`.

Note that L^AT_EX can only draw lines with `slope = x/y`, where `x` and `y` have integer values from -6 through 6.

`\linethickness{dimension}` Declares the thickness of horizontal and vertical lines in a picture environment to be `dimension`, which must be a positive length. It does not affect the thickness of slanted lines and circles, or the quarter circles drawn by `\oval` to form the corners of an oval.

`\makebox(width,height)[position]{...}` The `\makebox` command for the picture environment is similar to the normal `\makebox` command except that you must specify a `width` and `height` in multiples of `\unitlength`.

The optional argument, `[position]`, specifies the quadrant that your text appears in. You may select up to two of the following:

- `t` - Moves the item to the top of the rectangle
- `b` - Moves the item to the bottom
- `l` - Moves the item to the left
- `r` - Moves the item to the right

See `\makebox`.

`\multiput(x coord,y coord)(delta x,delta y){number of copies}{object}` The `\multiput` command can be used when you are putting the same object in a regular pattern across a picture.

`\oval(width,height)[portion]` The `\oval` command produces a rectangle with rounded corners. The optional argument, `[portion]`, allows you to select part of the oval.

- `t` - Selects the top portion
- `b` - Selects the bottom portion
- `r` - Selects the right portion
- `l` - Selects the left portion

Note that the *corners* of the oval are made with quarter circles with a maximum radius of 20 pt, so large *ovals* will look more like boxes with rounded corners.

`\put(x coord,y coord){ ... }` The `\put` command places the item specified by the mandatory argument at the given coordinates.

`\shortstack[position]{... \ \ ... \ \ ...}` The `\shortstack` command produces a stack of objects. The valid positions are:

- `r` - Moves the objects to the right of the stack
- `l` - Moves the objects to the left of the stack
- `c` - Moves the objects to the centre of the stack (default)

`\vector(x slope,y slope){length}` The `\vector` command draws a line with an arrow of the specified length and slope. The `x` and `y` values must lie between -4 and +4, inclusive.

quotation

```
\begin{quotation}
  text
\end{quotation}
```

The margins of the `quotation` environment are indented on the left and the right. The text is justified at both margins and there is paragraph indentation. Leaving a blank line between text produces a new paragraph.

quote

```
\begin{quote}
  text
\end{quote}
```

The margins of the `quote` environment are indented on the left and the right. The text is justified at both margins. Leaving a blank line between text produces a new paragraph.

tabbing

```
\begin{tabbing}
text \= more text \= still more text \= last text \\
second row \> \> more \\
.
.
.
\end{tabbing}
```

The `tabbing` environment provides a way to align text in columns. It works by setting tab stops and tabbing to them much the way you do with an ordinary typewriter.

It is best suited for cases where the width of each column is constant and known in advance.

This environment can be broken across pages, unlike the `tabular` environment.

The following commands can be used inside a `tabbing` environment:

<code>\=</code>	Sets a tab stop at the current position.
<code>\></code>	Advances to the next tab stop.
<code>\<</code>	This command allows you to put something to the left of the local margin without changing the margin. Can only be used at the start of the line.
<code>\+</code>	Moves the left margin of the next and all the following commands one tab stop to the right.
<code>\-</code>	Moves the left margin of the next and all the following commands one tab stop to the left.
<code>\'</code>	Moves everything that you have typed so far in the current column, i.e. everything from the most recent <code>\></code> , <code>\<</code> , <code>\'</code> , <code>\\</code> , or <code>\kill</code> command, to the right of the previous column, flush against the current column's tab stop.
<code>\‘</code>	Allows you to put text flush right against any tab stop, including tab stop 0. However, it can't move text to the right of the last column because there's no tab stop there. The <code>\‘</code> command moves all the text that follows it, up to the <code>\\</code> or <code>\end{tabbing}</code> command that ends the line, to the right margin of the tabbing environment. There must be no <code>\></code> or <code>\'</code> command between the <code>\‘</code> and the command that ends the line.
<code>\kill</code>	Sets tab stops without producing text. Works just like <code>\\</code> except that it throws away the current line instead of producing output for it. The effect of any <code>\=</code> , <code>\+</code> or <code>\-</code> commands in that line remain in effect.
<code>\pushtabs</code>	Saves all current tab stop positions. Useful for temporarily changing tab stop positions in the middle of a <code>tabbing</code> environment.
<code>\poptabs</code>	Restores the tab stop positions saved by the last <code>\pushtabs</code> .
<code>\a</code>	In a <code>tabbing</code> environment, the commands <code>\=</code> , <code>\'</code> and <code>\‘</code> do not produce accents as normal. Instead, the commands <code>\a=</code> , <code>\a'</code> and <code>\a‘</code> are used.

Table 1.4.: Commands in a `tabbing` environment

This example typesets a Pascal function in a traditional format:

```
\begin{tabbing}
function \= fact(n : integer) : integer;\
  \> begin \= \+ \
    \> if \= n > 1 then \+ \
      fact := n * fact(n-1) \- \
    else \+ \
      fact := 1; \- \
    end;\
\end{tabbing}
```

`table`

```
\begin{table}[placement]

  body of the table

\caption{table title}
\end{table}
```

Tables are objects that are not part of the normal text, and are usually *float*ed to a convenient place, like the top of a page. Tables will not be split between two pages.

The optional argument `[placement]` determines where L^AT_EX will try to place your table. There are four places where L^AT_EX can possibly put a float:

- **h** : Here - at the position in the text where the table environment appears.
- **t** : Top - at the top of a text page.
- **b** : Bottom - at the bottom of a text page.
- **p** : Page of floats - on a separate float page, which is a page containing no text, only floats.

The standard `report` and `article` classes use the default placement `[tbp]`.

The body of the table is made up of whatever text, L^AT_EX commands, etc., you wish. The `\caption` command allows you to title your table.

`tabular`

```
\begin{tabular}[pos]{cols}
column 1 entry & column 2 entry ... & column n entry \
.
.
.
\end{tabular}
```

or

```
\begin{tabular*}{width}[pos]{cols}
column 1 entry & column 2 entry ... & column n entry \
.
.
.
\end{tabular*}
```

These environments produce a box consisting of a sequence of rows of items, aligned vertically in columns. The mandatory and optional arguments consist of:

These commands can be used inside a `tabular` environment:

`\cline{i-j}` The `\cline` command draws horizontal lines across the columns specified, beginning in column *i* and ending in column *j*, which are identified in the mandatory argument.

width	Specifies the width of the <code>tabular*</code> environment. There must be rubber space between columns that can stretch to fill out the specified width.
pos	Specifies the vertical position; default is alignment on the centre of the environment. <ul style="list-style-type: none"> • <code>t</code> - align on top row • <code>b</code> - align on bottom row
cols	Specifies the column formatting. It consists of a sequence of the following specifiers, corresponding to the sequence of columns and intercolumn material. <ul style="list-style-type: none"> • <code>l</code> - A column of left-aligned items. • <code>r</code> - A column of right-aligned items. • <code>c</code> - A column of centred items. • <code> </code> - A vertical line the full height and depth of the environment. • <code>@{text}</code> - This inserts <code>text</code> in every row. An <code>@</code>-expression suppresses the intercolumn space normally inserted between columns; any desired space between the inserted text and the adjacent items must be included in text. An <code>\extracolsep{wd}</code> command in an <code>@</code>-expression causes an extra space of width <code>wd</code> to appear to the left of all subsequent columns, until countermanded by another <code>\extracolsep</code> command. Unlike ordinary intercolumn space, this extra space is not suppressed by an <code>@</code>-expression. An <code>\extracolsep</code> command can be used only in an <code>@</code>-expression in the <code>cols</code> argument. • <code>p{wd}</code> - Produces a column with each item typeset in a parbox of width <code>wd</code>, as if it were the argument of a <code>\parbox[t]{wd}</code> command. However, a <code>\\</code> may not appear in the item, except in the following situations: <ol style="list-style-type: none"> 1. inside an environment like <code>minipage</code>, <code>array</code>, or <code>tabular</code>. 2. inside an explicit <code>\parbox</code>. 3. in the scope of a <code>\centering</code>, <code>\raggedright</code>, or <code>\raggedleft</code> declaration. The latter declarations must appear inside braces or an environment when used in a <code>p</code>-column element. • <code>*{num}{cols}</code> - Equivalent to <code>num</code> copies of <code>cols</code>, where <code>num</code> is any positive integer and <code>cols</code> is any list of column-specifiers, which may contain another <code>*-expression</code>.

Table 1.5.: Commands used inside a `tabular` environment

`\hline` The `\hline` command will draw a horizontal line the width of the table. It's most commonly used to draw a line at the top, bottom, and between the rows of the table.

`\multicolumn` `\multicolumn{cols}{pos}{text}`

The `\multicolumn` is used to make an entry that spans several columns. The first mandatory argument, `cols`, specifies the number of columns to span. The second mandatory argument, `pos`, specifies the formatting of the entry; `c` for centred, `l` for flushleft, `r` for flushright. The third mandatory argument, `text`, specifies what text is to make up the entry.

`\vline` The `\vline` command will draw a vertical line extending the full height and depth of its row. An `\hfill` command can be used to move the line to the edge of the column. It can also be used in an `@`-expression.

`thebibliography`

```
\begin{thebibliography}{widest-label}
\bibitem[label]{cite_key}
.
.
.
\end{thebibliography}
```

The `thebibliography` environment produces a bibliography or reference list. In the `article` class, this reference list is labelled *References*; in the `report` class, it is labelled *Bibliography*.

- **widest-label**: Text that, when printed, is approximately as wide as the widest item label produces by the `\bibitem` commands.

`\bibitem[label]{cite_key}` The `\bibitem` command generates an entry labelled by `label`. If the `label` argument is missing, a number is generated as the `label`, using the `enumi` counter. The `cite_key` is any sequence of letters, numbers, and punctuation symbols not containing a comma. This command writes an entry on the `.aux` file containing `cite_key` and the item's `label`. When this `.aux` file is read by the `\begin{document}` command, the item's `label` is associated with `cite_key`, causing the reference to `cite_key` by a `\cite` command to produce the associated `label`.

`\cite[text]{key_list}` The `key_list` argument is a list of citation keys. This command generates an in-text citation to the references associated with the keys in `key_list` by entries on the `.aux` file read by the `\begin{document}` command.

The optional `text` argument will appear after the citation, i.e. `\cite[p.~2f.]{knuth}` might produce '[Knuth, p. 2]'

The package `cite.sty` allows a line break in the `\cite` reference and can sort numerical entries.

`overcite.sty` makes citations like footnotes.

`\nocite{key_list}` The `\nocite` command produces no text, but writes `key_list`, which is a list of one or more citation keys, on the `.aux` file. `\nocite{*}` uses all entries from the BIB_TE_X database.

Using BibT_EX If you use the BibT_EX program by Oren Patashnik (highly recommended if you need a bibliography of more than a couple of titles) to maintain your bibliography, you don't use the `thebibliography` environment. Instead, you include the lines

```
\bibliographystyle{style}
\bibliography{bibfile}
```

where `style` refers to a file `style.bst`, which defines how your citations will look.

<code>alpha</code>	Sorted alphabetically. Labels are formed from name of author and year of publication.
<code>plain</code>	Sorted alphabetically. Labels are numeric.
<code>unsrt</code>	Like <code>plain</code> , but entries are in order of citation.
<code>abbrv</code>	Like <code>plain</code> , but more compact labels.

Table 1.6.: Standard styles distributed with BibT_EX

In addition, numerous other BibT_EX style files exist tailored to the demands of various publications.

The argument to `\bibliography` refers to the file `bibfile.bib`, which should contain your database in BibT_EX format. Only the entries referred to via `\cite` and `\nocite` will be listed in the bibliography.

theorem

```
\begin{theorem}
theorem text
\end{theorem}
```

The `theorem` environment produces *Theorem x* in boldface followed by your theorem text.

titlepage

```
\begin{titlepage}
text
\end{titlepage}
```

The `titlepage` environment creates a title page, i.e. a page with no printed page number or heading. It also causes the following page to be numbered page one. Formatting the title page is left to you. The `\today` command comes in handy for title pages.

Note that you can use the `\maketitle` (see `\maketitle`) command to produce a standard title page.

`verbatim`

```
\begin{verbatim}
text
\end{verbatim}
```

The `verbatim` environment is a paragraph-making environment that gets L^AT_EX to print exactly what you type in. It turns L^AT_EX into a typewriter with carriage returns and blanks having the same effect that they would on a typewriter.

`\verb char literal_text char \verb*char literal_text char` Typesets `literal_text` exactly as typed, including special characters and spaces, using a typewriter (`\tt`) type style. There may be no space between `\verb` or `\verb*` and `char` (space is shown here only for clarity). The `*-form` differs only in that spaces are printed

`verse`

```
\begin{verse}
text
\end{verse}
```

The `verse` environment is designed for poetry, though you may find other uses for it.

The margins are indented on the left and the right. Separate the lines of each stanza with `\\`, and use one or more blank lines to separate the stanzas.

1.2.2.7. Footnotes

Footnotes can be produced in one of two ways. They can be produced with one command, the `\footnote` command. They can also be produced with two commands, the `\footnotemark` and the `\footnotetext` commands. See the specific command for information on why you would use one over the other.

`\footnote[number]{text}` The `\footnote` command places the numbered footnote `text` at the bottom of the current page. The optional argument, `number`, is used to change the default footnote number. This command can only be used in outer paragraph mode; i.e., you cannot use it in sectioning commands like `\chapter`, in figures, tables or in a `tabular` environment.

`\footnotemark` The `\footnotemark` command puts the footnote `number` in the text. This command can be used in inner paragraph mode. The text of the footnote is supplied by the `\footnotetext` command.

This command can be used to produce several consecutive footnote markers referring to the same footnote by using

```
\footnotemark[\value{footnote}]
```

after the first `\footnote` command.

`\footnotetext[number]{text}` The `\footnotetext` command produces the `text` to be placed at the bottom of the page. This command can come anywhere after the `\footnotemark` command. The `\footnotetext` command must appear in outer paragraph mode.

The optional argument, `number`, is used to change the default footnote number.

1.2.2.8. Lengths

A **length** is a measure of distance. Many L^AT_EX commands take a length as an argument.

```
\newlength \newlength{\gnat}
```

The `\newlength` command defines the mandatory argument, `\gnat`, as a **length** command with a value of 0in. An error occurs if a `\gnat` command already exists.

```
\setlength \setlength{\gnat}{length}
```

The `\setlength` command is used to set the value of a **length** command. The **length** argument can be expressed in any terms of length L^AT_EX understands, i.e., inches (in), millimetres (mm), points (pt), etc.

```
\addtolength \addtolength{\gnat}{length}
```

The `\addtolength` command increments *length command* by the amount specified in the **length** argument. It can be a negative amount.

```
\settodepth \settodepth{\gnat}{text}
```

The `\settodepth` command sets the value of a **length** command equal to the depth of the **text** argument.

```
\settoheight \settoheight{\gnat}{text}
```

The `\settoheight` command sets the value of a **length** command equal to the height of the **text** argument.

```
\settowidth \settowidth{\gnat}{text}
```

The `\settowidth` command sets the value of a **length** command equal to the width of the **text** argument.

```
Predefined lengths \width
```

```
\height
```

```
\depth
```

```
\totalheight
```

These length parameters can be used in the arguments of the box-making commands See Spaces & Boxes. They specify the natural width etc. of the text in the box. `\totalheight` equals `\height + \depth`. To make a box with the text stretched to double the natural size, e.g., say

```
\makebox[2\width]{Get a stretcher}
```

1.2.2.9. Letters

You can use L^AT_EX to typeset letters, both personal and business. The **letter** document class is designed to make a number of letters at once, although you can make just one if you so desire.

Your `.tex` source file has the same minimum commands as the other document classes, i.e., you must have the following commands as a minimum:

```
\documentclass{letter}
\begin{document}
... letters ...
\end{document}
```

Each letter is a **letter** environment, whose argument is the name and address of the recipient. For example, you might have:

```
\begin{letter}{Mr. Joe Smith\\ 2345 Princess St.\\ Edinburgh, EH1 1AA}
...
\end{letter}
```

The letter itself begins with the `\opening` command. The text of the letter follows. It is typed as ordinary L^AT_EX input. Commands that make no sense in a letter, like `\chapter`, do not work. The letter closes with a `\closing` command.

After the `\closing`, you can have additional material. The `\cc` command produces the usual *cc:* There's also a similar `\encl` command for a list of enclosures. With both these commands, use `\\` to separate the items.

These commands are used with the `letter` class:

```
\address \address{Return address}
```

The return address, as it should appear on the letter and the envelope. Separate lines of the address should be separated by `\\` commands. If you do not make an `\address` declaration, then the letter will be formatted for copying onto your organisation's standard letterhead. (See Overview, for details on your local implementation). If you give an `\address` declaration, then the letter will be formatted as a personal letter.

```
\cc \cc{Kate Schechter\\Rob McKenna}
```

Generate a list of other persons the letter was sent to. Each name is printed on a separate line.

```
\closing \closing{text}
```

The letter closes with a `\closing` command, i.e.,

```
\closing{Best Regards,}
```

```
\encl \encl{CV\\Certificates}
```

Generate a list of enclosed material.

```
\location \location{address}
```

This modifies your organisation's standard address. This only appears if the `firstpage` pagestyle is selected.

```
\makelabels \makelabels{number}
```

If you issue this command in the preamble, L^AT_EX will create a sheet of address labels. This sheet will be output before the letters.

```
\name \name{June Davenport}
```

Your name, used for printing on the envelope together with the return address.

```
\opening \opening{text}
```

The letter begins with the `\opening` command. The mandatory argument, `text`, is whatever text you wish to start your letter, i.e.,

```
\opening{Dear Joe,}
```

```
\ps \ps
```

Use this command before a postscript.

\signature \signature{Harvey Swick}

Your name, as it should appear at the end of the letter underneath the space for your signature. Items that should go on separate lines should be separated by `\\` commands.

\startbreaks \startbreaks

Used after a `\stopbreaks` command to allow page breaks again.

\stopbreaks \stopbreaks

Inhibit page breaks until a `\startbreaks` command occurs.

\telephone \telephone{number}

This is your telephone number. This only appears if the `firstpage` pagestyle is selected.

1.2.2.10. Line & Page Breaking

The first thing L^AT_EX does when processing ordinary text is to translate your input file into a string of glyphs and spaces. To produce a printed document, this string must be broken into lines, and these lines must be broken into pages. In some environments, you do the line breaking yourself with the `\\` command, but L^AT_EX usually does it for you.

`\\` \\[extra-space], \\[*extra-space]

The `\\` command tells L^AT_EX to start a new line. It has an optional argument, `extra-space`, that specifies how much extra vertical space is to be inserted before the next line. This can be a negative amount.

The `*` command is the same as the ordinary `\\` command except that it tells L^AT_EX not to start a new page after the line.

`\-` The `\-` command tells L^AT_EX that it may hyphenate the word at that point. L^AT_EX is very good at hyphenating, and it will usually find all correct hyphenation points. The `\-` command is used for the exceptional cases.

Note that when you insert `\-` commands in a word, the word will only be hyphenated at those points and not at any of the hyphenation points that L^AT_EX might otherwise have chosen.

\cleardoublepage The `\cleardoublepage` command ends the current page and causes all figures and tables that have so far appeared in the input to be printed. In a two-sided printing style, it also makes the next page a right-hand (odd-numbered) page, producing a blank page if necessary.

\clearpage The `\clearpage` command ends the current page and causes all figures and tables that have so far appeared in the input to be printed.

\enlargethispage \enlargethispage{size}

\enlargethispage*{size}

Enlarge the `\textheight` for the current page by the specified amount; e.g. `\enlargethispage{\baselineskip}` will allow one additional line.

The starred form tries to squeeze the material together on the page as much as possible. This is normally used together with an explicit `\pagebreak`.

\samepage \samepage

The object in the argument `{}` should be on the current page.

\fussy \fussy

This declaration (which is the default) makes T_EX more fussy about line breaking. This can avoid too much space between words, but may produce overfull boxes.

This command cancels the effect of a previous \sloppy command. \sloppy

\hyphenation \hyphenation{words}

The \hyphenation command declares allowed hyphenation points, where **words** is a list of words, separated by spaces, in which each hyphenation point is indicated by a - character.

\linebreak \linebreak[number]

The \linebreak command tells L^AT_EX to break the current line at the point of the command. With the optional argument, **number**, you can convert the \linebreak command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

The \linebreak command causes L^AT_EX to stretch the line so it extends to the right margin.

\newline The \newline command breaks the line right where it is. It can only be used in paragraph mode.

\newpage The \newpage command ends the current page. In contrast to \pagebreak it can produce a partly empty page, even when \flushbottom is active.

\nolinebreak \nolinebreak[number]

The \nolinebreak command prevents L^AT_EX from breaking the current line at the point of the command. With the optional argument, **number**, you can convert the \nolinebreak command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

\nopagebreak \nopagebreak[number]

The \nopagebreak command prevents L^AT_EX from breaking the current page at the point of the command. With the optional argument, **number**, you can convert the \nopagebreak command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

\pagebreak \pagebreak[number]

The \pagebreak command tells L^AT_EX to break the current page at the point of the command. With the optional argument, **number**, you can convert the \pagebreak command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

\sloppy \sloppy

This declaration makes T_EX less fussy about line breaking. This can prevent overfull boxes, but may leave too much space between words.

Lasts until a \fussy command is issued. \fussy.

1.2.2.11. Making Paragraphs

A paragraph is ended by one or more completely blank lines — lines not containing even a %. A blank line should not appear where a new paragraph cannot be started, such as in math mode or in the argument of a sectioning command.

\indent \indent

This produces a horizontal space whose width equals the width of the paragraph indentation. It is used to add paragraph indentation where it would otherwise be suppressed.

\noindent \noindent

When used at the beginning of the paragraph, it suppresses the paragraph indentation. It has no effect when used in the middle of a paragraph.

\par Equivalent to a blank line; often used to make command or environment definitions easier to read.

1.2.2.12. Margin Notes

The command `\marginpar[left]{right}` creates a note in the margin. The first line will be at the same height as the line in the text where the `\marginpar` occurs.

When you only specify the mandatory argument `right`, the text will be placed

- in the right margin for one-sided layout
- in the outside margin for two-sided layout
- in the nearest margin for two-column layout.

By issuing the command `\reversemarginpar`, you can force the marginal notes to go into the opposite (inside) margin.

When you specify both arguments, `left` is used for the left margin, and `right` is used for the right margin.

The first word will normally not be hyphenated; you can enable hyphenation by prefixing the first word with a `\hspace{0pt}` command.

1.2.2.13. Math Formulae

<code>math</code>	For Formulae that appear right in the text.
<code>displaymath</code>	For Formulae that appear on their own line.
<code>equation</code>	The same as the <code>displaymath</code> environment except that it adds an equation number in the right margin.

Table 1.7.: Three environments that put L^AT_EX in math mode

The `math` environment can be used in both paragraph and LR mode, but the `displaymath` and `equation` environments can be used only in paragraph mode. The `math` and `displaymath` environments are used so often that they have the following short forms:

`\(...\)` instead of `\begin{math}...\end{math}`

`\[...\]` instead of `\begin{displaymath}...\end{displaymath}`

In fact, the `math` environment is so common that it has an even shorter form:

`$... $` instead of `\(...\)`

Subscripts & Superscripts To get an expression *exp* to appear as a subscript, you just type `_ {exp}`. To get *exp* to appear as a superscript, you type `^ {exp}`. L^AT_EX handles superscripted superscripts and all of that stuff in the natural way. It even does the right thing when something has both a subscript and a superscript.

Math Symbols

Spacing in Math Mode In a `math` environment, L^AT_EX ignores the spaces you type and puts in the spacing that it thinks is best. L^AT_EX formats mathematics the way it's done in mathematics texts. If you want different spacing, L^AT_EX provides the following four commands for use in math mode:

1. `\;` - a thick space
2. `\:` - a medium space
3. `\,` - a thin space
4. `\!` - a negative thin space

<code>\cdots</code>	Produces a horizontal ellipsis where the dots are raised to the centre of the line.
<code>\ddots</code>	Produces a diagonal ellipsis.
<code>\frac{num}{den}</code>	Produces the fraction <code>num</code> divided by <code>den</code> .
<code>\ldots</code>	Produces an ellipsis. This command works in any mode, not just math mode.
<code>\overbrace{text}</code>	Generates a brace over <code>text</code> .
<code>\overline{text}</code>	Causes the argument <code>text</code> to be overlined.
<code>\sqrt[root]{arg}</code>	Produces the square root of its argument. The optional argument, <code>root</code> , determines what root to produce, i.e., the cube root of <code>x+y</code> would be typed as <code>\sqrt[3]{x+y}</code> .
<code>\underbrace{text}</code>	Generates <code>text</code> with a brace underneath.
<code>\underline{text}</code>	Causes the argument <code>text</code> to be underlined. This command can also be used in paragraph and LR modes.
<code>\vdots</code>	Produces a vertical ellipsis.
<code>\ensuremath{}</code>	It's argument is set in math mode. This is needed for own definitions.

Table 1.8.: Math Miscellany

Math Miscellany

1.2.2.14. Modes

When L^AT_EX is processing your input text, it is always in one of three modes:

- paragraph mode
- Math mode
- Left-to-right mode, called LR mode for short

L^AT_EX changes mode only when it goes up or down a staircase to a different level, though not all level changes produce mode changes. Mode changes occur only when entering or leaving an environment, or when L^AT_EX is processing the argument of certain text-producing commands.

Paragraph mode is the most common; it's the one L^AT_EX is in when processing ordinary text. In that mode, L^AT_EX breaks your text into lines and breaks the lines into pages. L^AT_EX is in *math mode* when it's generating a mathematical formula. In *LR mode*, as in paragraph mode, L^AT_EX considers the output that it produces to be a string of words with spaces between them. However, unlike paragraph mode, L^AT_EX keeps going from left to right; it never starts a new line in LR mode. Even if you put a hundred words into an `\mbox`, L^AT_EX would keep typesetting them from left to right inside a single box, and then complain because the resulting box was too wide to fit on the line.

L^AT_EX is in LR mode when it starts making a box with an `\mbox` command. You can get it to enter a different mode inside the box - for example, you can make it enter math mode to put a formula in the box. There are also several text-producing commands and environments for making a box that put L^AT_EX in paragraph mode. The box made by one of these commands or environments will be called a **parbox**. When L^AT_EX is in paragraph mode while making a box, it is said to be in *inner paragraph mode*. Its normal paragraph mode, which it starts out in, is called *outer paragraph mode*.

1.2.2.15. Page Styles

The `\documentclass` command determines the size and position of the page's head and foot. The page style determines what goes in them.

\maketitle \maketitle

The `\maketitle` command generates a title on a separate title page - except in the `article` class, where the title normally goes at the top of the first page. Information used to produce the title is obtained from the following declarations:

See Page Styles for the commands to give the information.

\author \author{names}

The `\author` command declares the author(s), where `names` is a list of authors separated by `\and` commands. Use `\\` to separate lines within a single author's entry - for example, to give the author's institution or address.

\date \date{text}

The `\date` command declares `text` to be the document's date. With no `\date` command, the current date is used.

\thanks \thanks{text}

The `\thanks` command produces a `\footnote` to the title.

\title \title{text}

The `\title` command declares `text` to be the title. Use `\\` to tell L^AT_EX where to start a new line in a long title.

\pagenumbering \pagenumbering{num_style}

Specifies the style of page numbers. Possible values of `num_style` are:

- `arabic` - Arabic numerals
- `roman` - Lowercase Roman numerals
- `Roman` - Uppercase Roman numerals
- `alph` - Lowercase letters
- `Alph` - Uppercase letters

\pagestyle \pagestyle{option}

The `\pagestyle` command changes the style from the current page on throughout the remainder of your document.

The valid options are:

- `plain` - Just a plain page number.
- `empty` - Produces empty heads and feet - no page numbers.
- `headings` - Puts running headings on each page. The document style specifies what goes in the headings.
- `myheadings` - You specify what is to go in the heading with the `\markboth` or the `\markright` commands.

\markboth`\markboth{left head}{right head}`

The `\markboth` command is used in conjunction with the page style `myheadings` for setting both the left and the right heading. You should note that a *left-hand heading* is generated by the last `\markboth` command before the end of the page, while a *right-hand heading* is generated by the first `\markboth` or `\markright` that comes on the page if there is one, otherwise by the last one before the page.

\markright

```
\markright{right head}
```

The `\markright` command is used in conjunction with the page style `myheadings` for setting the right heading, leaving the left heading unchanged. You should note that a *left-hand heading* is generated by the last `\markboth` command before the end of the page, while a *right-hand heading* is generated by the first `\markboth` or `\markright` that comes on the page if there is one, otherwise by the last one before the page.

```
\thispagestyle \thispagestyle{option}
```

The `\thispagestyle` command works in the same manner as the `\pagestyle` command except that it changes the style for the current page only.

1.2.2.16. Sectioning

Sectioning commands provide the means to structure your text into units.

- `\part`
- `\chapter` (report and book class only)
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`

All sectioning commands take the same general form, i.e.,

```
\chapter[optional]{title}
```

In addition to providing the heading in the text, the mandatory argument of the sectioning command can appear in two other places:

1. The table of contents
2. The running head at the top of the page

You may not want the same thing to appear in these other two places as appears in the text heading. To handle this situation, the sectioning commands have an `optional` argument that provides the text for these other two purposes.

All sectioning commands have `*`-forms that print a *title*, but do not include a number and do not make an entry in the table of contents.

\appendix

The `\appendix` command changes the way sectional units are numbered. The `\appendix` command generates no text and does not affect the numbering of parts. The normal use of this command is something like

```
\chapter{The First Chapter}
...
\appendix
\chapter{The First Appendix}
```

1.2.2.17. Spaces & Boxes

All the predefined length parameters See Predefined lengths can be used in the arguments of the box-making commands.

\dotfill The `\dotfill` command produces a *rubber length* that produces dots instead of just spaces.

\hfill The `\hfill` fill command produces a *rubber length* which can stretch or shrink horizontally. It will be filled with spaces.

\hrulefill The `\hrulefill` fill command produces a *rubber length* which can stretch or shrink horizontally. It will be filled with a horizontal rule.

\hspace `\hspace{length}`, `\hspace*{length}`

The `\hspace` command adds horizontal space. The length of the space can be expressed in any terms that L^AT_EX understands, i.e., points, inches, etc. You can add negative as well as positive space with an `\hspace` command. Adding negative space is like backspacing.

L^AT_EX removes horizontal space that comes at the end of a line. If you don't want L^AT_EX to remove this space, include the optional `*` argument. Then the space is never removed.

\addvspace `\addvspace{length}`

The `\addvspace` command normally adds a vertical space of height `length`. However, if vertical space has already been added to the same point in the output by a previous `\addvspace` command, then this command will not add more space than needed to make the natural length of the total vertical space equal to `length`.

\bigskip The `\bigskip` command is equivalent to `\vspace{bigskipamount}` where `bigskipamount` is determined by the document class.

\medskip The `\medskip` command is equivalent to `\vspace{medskipamount}` where `medskipamount` is determined by the document class.

\smallskip `\smallskip`

The `\smallskip` command is equivalent to `\vspace{smallskipamount}` where `smallskipamount` is determined by the document class.

\vfill The `\vfill` fill command produces a rubber length which can stretch or shrink vertically.

\vspace `\vspace{length}`, `\vspace*{length}`

The `\vspace` command adds vertical space. The length of the space can be expressed in any terms that L^AT_EX understands, i.e., points, inches, etc. You can add negative as well as positive space with an `\vspace` command.

L^AT_EX removes vertical space that comes at the end of a page. If you don't want L^AT_EX to remove this space, include the optional `*` argument. Then the space is never removed.

\fbox `\fbox{text}`

The `\fbox` command is exactly the same as the `\mbox` command, except that it puts a frame around the outside of the box that it creates.

\framebox `\framebox[width][position]{text}`

The `\framebox` command is exactly the same as the `\makebox` command, except that it puts a frame around the outside of the box that it creates.

The `\framebox` command produces a rule of thickness `\fboxrule`, and leaves a space `\fboxsep` between the rule and the contents of the box.

lrbox `\begin{lrbox}{cmd} text \end{lrbox}`

This is the environment form of `\sbox`.

The text inside the environment is saved in the box `cmd`, which must have been declared with `\newsavebox`.

makebox `\makebox[width][position]{text}`

The `\makebox` command creates a box just wide enough to contain the `text` specified. The width of the box is specified by the optional `width` argument. The position of the text within the box is determined by the optional `position` argument.

- `c` — centred (default)
- `l` — flushleft
- `r` — flushright
- `s` — stretch from left to right margin. The text must contain stretchable space for this to work.

See `\makebox` (picture).

mbox `\mbox{text}`

The `\mbox` command creates a box just wide enough to hold the text created by its argument.

Use this command to prevent text from being split across lines.

newsavebox `\newsavebox{cmd}`

Declares `cmd`, which must be a command name that is not already defined, to be a bin for saving boxes.

parbox `\parbox[position][height][inner-pos]{width}{text}`

A `parbox` is a box whose contents are created in `paragraph` mode. The `\parbox` has two mandatory arguments:

- `width` - specifies the width of the parbox, and
- `text` - the text that goes inside the parbox.

L^AT_EX will position a `parbox` so its centre lines up with the centre of the text line. The optional `position` argument allows you to line up either the top or bottom line in the parbox (default is top).

If the `height` argument is not given, the box will have the natural height of the text.

The `inner-pos` argument controls the placement of the text inside the box. If it is not specified, `position` is used.

- `t` — text is placed at the top of the box.
- `c` — text is centred in the box.
- `b` — text is placed at the bottom of the box.
- `s` — stretch vertically. The text must contain vertically stretchable space for this to work.

A `\parbox` command is used for a parbox containing a small piece of text, with nothing fancy inside. In particular, you shouldn't use any of the paragraph-making environments inside a `\parbox` argument. For larger pieces of text, including ones containing a paragraph-making environment, you should use a `minipage` environment See `minipage`.

raisebox `\raisebox{distance}[extend-above][extend-below]{text}`

The `\raisebox` command is used to raise or lower text. The first mandatory argument specifies how high the text is to be raised (or lowered if it is a negative amount). The text itself is processed in `LR mode`.

Sometimes it's useful to make L^AT_EX think something has a different size than it really does - or a different size than L^AT_EX would normally think it has. The `\raisebox` command lets you tell L^AT_EX how tall it is.

The first optional argument, `extend-above`, makes L^AT_EX think that the text extends above the line by the amount specified. The second optional argument, `extend-below`, makes L^AT_EX think that the text extends below the line by the amount specified.

`\rule` `\rule[raise-height]{width}{thickness}`

The `\rule` command is used to produce horizontal lines. The arguments are defined as follows:

- `raise-height` - specifies how high to raise the rule (optional)
- `width` - specifies the length of the rule (mandatory)
- `thickness` - specifies the thickness of the rule (mandatory)

`\savebox` `\savebox{cmd}[width][pos]{text}`

This command typeset `text` in a box just as for `\makebox`. However, instead of printing the resulting box, it saves it in bin `cmd`, which must have been declared with `\newsavebox`.

`\sbox` `\sbox{cmd}\{text}`

This commands typesets `text` in a box just as for `\mbox`. However, instead of printing the resulting box, it saves it in `cmd`, which must have been declared with `\newsavebox`.

`\usebox` `\usebox{cmd}`

Prints the box most recently saved in bin `cmd` by a `\savebox` command.

1.2.2.18. Special Characters

The following characters play a special role in L^AT_EX and are called *special printing characters*, or simply *special characters*.

\$ % & ~ _ ^ \ { }

Whenever you put one of these special characters into your file, you are doing something special. If you simply want the character to be printed just as any other letter, include a `\` in front of the character. For example, `\$` will produce \$ in your output.

One exception to this rule is the `\` itself because `\\` has its own special meaning. A `\` is produced by typing `$$backslash$` in your file.

Also, `\~` means ‘place a tilde accent over the following letter’, so you will probably want to use `\verb` instead.

In addition, you can access any character of a font once you know its number by using the `\symbol` command. For example, the character used for displaying spaces in the `\verb*` command has the code decimal 32, so it can be typed as `\symbol{32}`.

You can also specify octal numbers with `'` or hexadecimal numbers with `"`, so the previous example could also be written as `\symbol{'40}` or `\symbol{"20}`.

1.2.2.19. Special Symbols

Accents on Characters The rules differ somewhat depending whether you are in text mode, math modes, or the tabbing environment.

Text Mode The following accents may be placed on letters. Although `o` is used in most of the example, the accents may be placed on any letter. Accents may even be placed above a *missing* letter; for example, `\~{}` produces a tilde over a blank space.

The following commands may be used only in paragraph or LR mode:

- `\‘{o}` produces a grave accent, \grave{o}
- `\’{o}` produces an acute accent, \acute{o}
- `\ˆ{o}` produces a circumflex, \hat{o}
- `\" {o}` produces an umlaut or dieresis, \ddot{o}

- `\H{o}` produces a long Hungarian umlaut
- `\~{o}` produces a tilde, \tilde{o}
- `\c{c}` produces a cedilla, $\ç$
- `\={o}` produces a macron accent (a bar over the letter)
- `\b{o}` produces a bar under the letter
- `\. {o}` produces a dot over the letter
- `\d{o}` produces a dot under the letter
- `\u{o}` produces a breve over the letter
- `\v{o}` produces a v over the letter
- `\t{oo}` produces a *tie* (inverted u) over the two letters

Note that the letters i and j require special treatment when they are given accents because it is often desirable to replace the dot with the accent. For this purpose, the commands `\i` and `\j` can be used to produce dotless letters.

For example,

- `\^{i}` should be used for i , circumflex, \hat{i}
- `\" {i}` should be used for i , umlaut, \ddot{i}

Math Mode Several of the above and some similar accents can also be produced in math mode. The following commands may be used *only* in math mode:

- `\hat{o}` is similar to the circumflex (cf. `\^`)
- `\widehat{oo}` is a wide version of `\hat` over several letters
- `\check{o}` is a vee or check (cf. `\v`)
- `\tilde{o}` is a tilde (cf. `\~`)
- `\widetilde{oo}` is a wide version of `\tilde` over several letters
- `\acute{o}` is an acute accent (cf. `\´`)
- `\grave{o}` is a grave accent (cf. `>\´`)
- `\dot{o}` is a dot over the letter (cf. `\.`)
- `\ddot{o}` is a double dot over the letter
- `\breve{o}` is a breve (cf. `\u`)
- `\bar{o}` is a macron (cf. `\=`)
- `\vec{o}` is a vector (arrow) over the letter

Tabbing Environment Some of the accent marks used in running text have other uses in the Tabbing Environment. In that case they can be created with the following command:

- `>\a´` for an acute accent
- `\a´` for a grave accent
- `\a=` for a macron accent

Arrows L^AT_EX provides commands for all sorts of arrows. The following commands are used only in math mode.

In general the command names are created from `left`, `right`, `up`, or `down`. `leftright` gives a double headed arrow. Prefacing with `long` gives a longer arrow. Capitalizing the first letter gives a double-shanked arrow.

Examples are:

- `\leftarrow`
- `\Leftarrow`
- `\longleftarrow`
- `\Longleftarrow`
- `\rightarrow`
- `\Rightarrow`
- `\longrightarrow`
- `\Longrightarrow`
- `\leftrightarrow`
- `\Leftrightarrow`
- `\longleftrightarrow`
- `\uparrow`

- `\downarrow`
- `\Uparrow`
- `>\nearrow` points from southwest to northeast
- `\searrow` points from northwest to southeast
- `\swarrow` points from northeast to southwest
- `\nwarrow` points from southeast to northwest

The `\stackrel` command is useful for putting things over or under your arrow.

See also:

- Math Symbols
- Binary and Relational Operators
- Greek Letters
- Miscellaneous Symbols
- Variable Size Math Symbols

Binary and Relational Operators Some *math* symbols are obtained by typing the corresponding keyboard character. Examples include

+ - = < >

NOTE

plus, minus, and equal sign may be used in either text or math mode, but < and > are math mode only (they produce inverted exclamation and question marks, respectively, in text mode).

The following commands may be used only in math mode:

- `\pm` plus or minus sign
- `\mp` minus or plus sign
- `\times` times (an "x")
- `\div` divided by sign
- `\ast` an asterisk (centered)
- `\star` a five-point star (centered)
- `\bullet` a bullet
- `\circ` an open bullet
- `\cdot` a centered dot
- `\leq` less than or equal to
- `\ll` much less than
- `\subset` is a subset of
- `\geq` greater than or equal to
- `\gg` much greater than
- `\equiv` equivalence symbol
- `\sim` similar to
- `\simeq` similar or equal to
- `\approx` approximately equal to
- `\neq` not equal to
- `\perp` "perpendicular to" symbols
- `\propto` proportional to

Note that *greater than* and *less than* are obtained simply by entering > and <.

A slash, indicating *not* can be placed through a symbol (or a letter) with the `\not` command. For example, not less than is `\not<` and not less than or equal to is `\not\leq`. If the slash isn't properly positioned it can be moved by putting a math mode spacing command between the `\not` and the symbol.

Delimiters Delimiters are objects which act logically like parentheses. These can be used only in math mode.

The delimiters recognized by L^AT_EX include:

- ((left parenthesis)

-) (right parenthesis)
- [(left bracket)
-] (right bracket)
- \{ (left brace)
- \} (right brace)
- | (vertical line)
- \vert (vertical line)
- \!| (double vertical lines)
- \Vert (double vertical lines)
- / (slash)
- \backslash (backslash)
- \langle (left angle bracket)
- \rangle (right angle bracket)
- \uparrow (uparrow)
- \downarrow (down arrow)
- \updownarrow (up/down arrow)

Making Delimiters the Right Size Delimiters in formulas should be big enough to *fit* around the formulas they delimit (for example around arrays). To obtain *stretchable* delimiters (L^AT_EX makes them the appropriate size) type a `\left` or `\right` command before the delimiter. `\left` and `\right` commands must come in matching pairs, although the delimiters themselves need not be the same. Thus, `\left \{ ... \right [` produces a legal pair. In cases where only one delimiter is desired, it is possible to make the matching delimiter *invisible* by typing a period (.) after the command, i.e., `\left.` or `\right.`

In an `eqnarray` environment the matching `\left` and `\right` cannot be split between lines and it may be necessary to use an *invisible* `\right.` and `\left.` to terminate and begin parts on different lines. In this case a second problem may arise, since the size of the delimiters will be chosen only for the *local part*, so that the size of the *visible left* and *right* delimiters might not match. The solution is to trick L^AT_EX into thinking that both parts have the same vertical height. This can be done by placing a strut, that is a zero-width `\rule`. It can also be accomplished with the `\vphantom` command, which I have not found documented, but which appears to work.

`\vphantom{construct}` creates a zero-width object with the height of `construct`. The argument can contain things such as `\frac` or the *variable size math symbols* and should be chosen according to what is in the section with the delimiter you want to match.

Some Examples

A six-j symbol

```
\[ \left\{
\begin{array}{ccc}
a & b & c \\
d & e & f
\end{array}
\right.\} \]
```

This should be displayed something like (insofar as it can be rendered in "ascii art"):

```
( a b c )
-       -
( d e f )
```

Note that the `\[... \]` set this off as *Display Math*, and that the *Array Environment* is used to generate the three centered columns inside the braces.

A "multiple choice" equation

```
\[ f(x) =
  \left\{ \begin{array}{l}
    0, x < 0 \\
    1, x = 0 \\
    2, x > 0 \end{array} \right. \]
```

will be displayed as

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x = 0 \\ 2, & x > 0 \end{cases}$$

Note that the *invisible* `\right` delimiter is specified using a *period*.

Ellipsis Ellipses (three dots) can be produced by the following commands

- `\ldots` horizontally at bottom of line
- `\cdots` horizontally center of line (math mode only)
- `\ddots` diagonal (math mode only)
- `\vdots` vertical (math mode only)

Greek Letters These commands may be used only in math mode:

- `\alpha`
- `\beta`
- `\gamma`
- `\delta`
- `\epsilon`
- `\varepsilon` (variation, script-like)
- `\zeta`
- `\eta`
- `\theta`
- `\vartheta` (variation, script-like)
- `\iota`
- `\kappa`
- `\lambda`
- `\mu`
- `\nu`
- `\xi`
- `\pi`
- `\varpi` (variation)
- `\rho`
- `\varrho` (variation, with the tail)
- `\sigma`
- `\varsigma` (variation, script-like)
- `\tau`
- `\upsilon`
- `\phi`
- `\varphi` (variation, script-like)
- `\chi`
- `\psi`
- `\omega`

Capital letters:

- `\Gamma`
- `\Delta`
- `\Theta`
- `\Lambda`
- `\Xi`
- `\Pi`
- `\Sigma`

- `\Upsilon`
- `\Phi`
- `\Psi`
- `\Omega`

See also:

- Math Symbols
- Accents
- Miscellaneous Symbols

Miscellaneous Symbols

Some symbols for math The following symbols are also used only in math mode

- `\aleph` Hebrew aleph
- `\hbar` h-bar, Planck's constant
- `\imath` variation on *i*; no dot
- `\jmath` variation on *j*; no dot
- `\ell` script (loop) *l*
- `\wp` fancy script lowercase *P*
- `\Re` script capital *R* (*Real*)
- `\Im` script capital *I* (*Imaginary*)
- `\prime` prime (also obtained by typing `'`)
- `\nabla` inverted capital Delta
- `\surd` radical (square root) symbol
- `\angle` angle symbol
- `\forall` *for all* (inverted A)
- `\exists` *exists* (left-facing E)
- `\partial` partial derivative symbol
- `\infty` infinity symbol
- `\triangle` open triangle symbol
- `\Box` open square
- `\Diamond` open diamond
- `\flat` music: flat symbol
- `\natural` music: natural symbol
- `\clubsuit` playing cards: club suit symbol
- `\diamondsuit` playing cards: diamond suit symbol
- `\heartsuit` playing cards: heart suit symbol
- `\spadesuit` playing cards: space suit symbol

Some Other Symbols The following symbols can be used in any mode:

- `\dag` dagger
- `\ddag` double dagger
- `\S` section symbol
- `\P` paragraph symbol
- `\copyright` copyright symbol
- `\pounds` British pound sterling symbol

Calligraphic Style Letters Twenty-six calligraphic letters are provided (the upper case alphabet). These can only be used in math mode.

In L^AT_EX 2.09 they are produced with the `\cal` declaration:

`\cal A`

In L^AT_EX 2_ε they are obtained with the `\mathcal` command:

`\mathcal{A}`

Math Functions Functions like *log* or *cos* are normally used in math mode. However, inmath mode strings of letters are treated as a product of variables, which would normally be displayed in math italics rather than a text font which would be appropriate for these functions. To get proper display of such functions they are generated with L^AT_EX commands.

Some of these commands are:

- `\arccos`
- `\arcsin`
- `\arctan`
- `\cos`
- `\cosh`
- `\cot`
- `\coth`
- `\csc`
- `\deg`
- `\det`
- `\dim`
- `\exp`
- `\gcd`
- `\hom`
- `\inf`
- `\ker`
- `\lg`
- `\lim`
- `\liminf`
- `\limsup`
- `\ln`
- `\log`
- `\max`
- `\min`
- `\sec`
- `\sin`
- `\sinh`
- `\sup`
- `\tan`
- `\tanh`

Two commands are provided for the "modulus" function

- `\bmod`
- `\pmod{}`

The former would be used to write "a mod b" as `\bmod b` and the latter, which requires an argument that is displayed in parentheses, would be used to write "n = i (mod j)" as `n = i \pmod{j}`

Variable Size Math Symbols The size of some mathematical symbols, notably summation signs, product signs, and integral signs, depends on the environment in which they appear (i.e., `displaymath` as opposed to `math` environments).

These include:

- `\sum` a summation sign (capital sigma)
- `\prod` a product (capital pi)
- `\coprod` a coproduct (inverted capital pi)
- `\int` an integral sign
- `\oint` a surface (circular) integral sign
- `\bigcup` big "U"
- `\bigcap` big inverted "U"
- `\bigvee` big "V"
- `\bigwedge` big inverted "V"
- `\bigodot` big "O" with dot at center
- `\bigotimes` big "O" with cross inside

- `\bigoplus` big "O" with a + inside
- `\biguplus` big "U" with a + inside

The `\sqrt` command also produces a variable size symbol appropriate for the size of the radicand argument.

The "limits" associated with these symbols are entered as *subscripts* for entries appearing below the symbol and as *superscripts* for entries appearing above the symbol. For example the sum from $n=0$ to infinity of x^n would be entered as

```
\sum_{n=0}^{\infty} x_{n}
```

The actual placement of the limits depends on whether this is in `\displaymath` mode in which case they are placed below/above or in `math` mode in running text in which case they are placed as regular subscripts and superscripts.

Note that it is possible to treat several of these symbols (a common example would be a double sum) as a single symbol for placing limits above and/or below by using the `\mathop` command.

Hats and *tildes* over symbols which stretch (as best they can) to the correct size for their arguments are produced by `\widehat` and `\widetilde`

1.2.2.20. Splitting the Input

A large document requires a lot of input. Rather than putting the whole input in a single large file, it's more efficient to split it into several smaller ones. Regardless of how many separate files you use, there is one that is the root file; it is the one whose name you type when you run L^AT_EX.

```
\include \include{file}
```

The `\include` command is used in conjunction with the `\includeonly` command for selective inclusion of files. The `file` argument is the first name of a file, denoting `file.tex`. If `file` is one of the file names in the file list of the `\includeonly` command or if there is no `\includeonly` command, the `\include` command is equivalent to

```
\clearpage \input{file} \clearpage
```

except that if the file `file.tex` does not exist, then a warning message rather than an error is produced. If the file is not in the file list, the `\include` command is equivalent to `\clearpage`.

The `\include` command may not appear in the preamble or in a file read by another `\include` command.

```
\includeonly \includeonly{file_list}
```

The `\includeonly` command controls which files will be read in by an `\include` command. `file_list` should be a comma-separated list of filenames. Each filename must match exactly a filename specified in a `\include` command. This command can only appear in the preamble.

```
\input \input{file}
```

The `\input` command causes the indicated `file` to be read and processed, exactly as if its contents had been inserted in the current file at that point. The file name may be a complete file name with extension or just a first name, in which case the file `file.tex` is used.

1.2.2.21. Starting & Ending

Your input file must contain the following commands as a minimum:

```
\documentclass{class}
\begin{document}
... your text goes here ...
\end{document}
```

where the `class` selected is one of the valid classes for L^AT_EX. See Document Classes (and see Overview), for details of the various document classes available locally.

You may include other L^AT_EX commands between the `\documentclass` and the `\begin{document}` commands (i.e., in the ‘preamble’).

1.2.2.22. Table of Contents

A table of contents is produced with the `\tableofcontents` command. You put the command right where you want the table of contents to go; L^AT_EX does the rest for you. It produces a heading, but it does not automatically start a new page. If you want a new page after the table of contents, include a `\newpage` command after the `\tableofcontents` command.

There are similar commands `\listoffigures` and `\listoftables` for producing a list of figures and a list of tables, respectively. Everything works exactly the same as for the table of contents.

NOTE: If you want any of these items to be generated, you cannot have the `\nofiles` command in your document.

`\addcontentsline` `\addcontentsline{file}{sec_unit}{entry}`

The `\addcontentsline` command adds an entry to the specified list or table where:

- `file` is the extension of the file on which information is to be written: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).
- `sec_unit` controls the formatting of the entry. It should be one of the following, depending upon the value of the file argument:
 1. `toc` — the name of the sectional unit, such as part or subsection.
 2. `lof` — figure
 3. `lot` — table
- `entry` is the text of the entry.

`\addtocontents` `\addtocontents{file}{text}`

The `\addtocontents` command adds text (or formatting commands) directly to the file that generates the table of contents or list of figures or tables.

- `file` is the extension of the file on which information is to be written: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).
- `text` is the information to be written.

1.2.2.23. Terminal Input/Output

`\typein` `\typein[cmd]{msg}`

Prints `msg` on the terminal and causes L^AT_EX to stop and wait for you to type a line of input, ending with return. If the `cmd` argument is missing, the typed input is processed as if it had been included in the input file in place of the `\typein` command. If the `cmd` argument is present, it must be a command name. This command name is then defined or redefined to be the typed input.

`\typeout` `\typeout{msg}`

Prints `msg` on the terminal and in the log file. Commands in `msg` that are defined with `\newcommand` or `\renewcommand` are replaced by their definitions before being printed.

L^AT_EX’s usual rules for treating multiple spaces as a single space and ignoring spaces after a command name apply to `msg`. A `\space` command in `msg` causes a single space to be printed. A `^^J` in `msg` prints a newline.

1.2.2.24. Typefaces

The typeface is specified by giving the *size* and *style*. A typeface is also called a *font*.

\Styles The following type style commands are supported by L^AT_EX.

These commands are used like `\textit{italics text}`. The corresponding command in parenthesis is the *declaration form*, which takes no arguments. The scope of the declaration form lasts until the next type style command or the end of the current group.

The declaration forms are cumulative; i.e., you can say `\sffamily\bfseries` to get sans serif boldface.

You can also use the environment form of the declaration forms; e.g. `\begin{ttfamily}... \end{ttfamily}`.

<code>\textrm (\rmfamily)</code>	Roman.
<code>\textit (\itshape)</code> , <code>\emph</code>	Emphasis (toggles between <code>\textit</code> and <code>\textrm</code>).
<code>\textmd (\mdseries)</code>	Medium weight (default). The opposite of boldface.
<code>\textbf (\bfseries)</code>	Boldface.
<code>\textup (\upshape)</code>	Upright (default). The opposite of slanted.
<code>\textsl (\slshape)</code>	Slanted.
<code>\textsf (\sffamily)</code>	Sans serif.
<code>\textsc (\scshape)</code>	Small caps.
<code>\texttt (\ttfamily)</code>	Typewriter.
<code>\textnormal (\normalfont)</code>	Main document font.
<code>\mathrm</code>	Roman, for use in math mode.
<code>\mathbf</code>	Boldface, for use in math mode. For bold symbols or complete equations take a look at <code>\boldsymbol</code> and <code>\bm</code> .
<code>\mathsf</code>	Sans serif, for use in math mode.
<code>\mathtt</code>	Typewriter, for use in math mode.
<code>\mathit</code>	Italics, for use in math mode, e.g. variable names with several letters.
<code>\mathnormal</code>	For use in math mode, e.g. inside another type style declaration.
<code>\mathcal</code>	‘Calligraphic’ letters, for use in math mode.

Table 1.9.: Typefaces

In addition, the command `\mathversion{bold}` can be used for switching to bold letters and symbols in formulas. `\mathversion{normal}` restores the default.

Sizes The following standard type size commands are supported by L^AT_EX.

The commands as listed here are "declaration forms". The scope of the declaration form lasts until the next type style command or the end of the current group.

You can also use the environment form of these commands; e.g. `\begin{tiny}... \end{tiny}`.

<code>\tiny</code>	
<code>\scriptsize</code>	
<code>\footnotesize</code>	
<code>\small</code>	
<code>\normalsize</code>	(default)
<code>\large</code>	
<code>\Large</code>	
<code>\LARGE</code>	
<code>\huge</code>	
<code>\Huge</code>	

Table 1.10.: Standard type size commands

Low-level font commands These commands are primarily intended for writers of macros and packages. The commands listed here are only a subset of the available ones. For full details, you should consult Chapter 7 of *The L^AT_EX Companion*.

<code>\fontencoding{enc}</code>	Select font encoding. Valid encodings include OT1 and T1.
<code>\fontfamily{family}</code>	Select font family. Valid families include: <ul style="list-style-type: none"> • <code>cmr</code> for Computer Modern Roman • <code>cmss</code> for Computer Modern Sans Serif • <code>cmtt</code> for Computer Modern Typewriter and numerous others.
<code>\fontseries{series}</code>	Select font series. Valid series include: <ul style="list-style-type: none"> • <code>m</code> Medium (normal) • <code>b</code> Bold • <code>c</code> Condensed • <code>bc</code> Bold condensed • <code>bx</code> Bold extended and various other combinations.
<code>\fontshape{shape}</code>	Select font shape. Valid shapes are: <ul style="list-style-type: none"> • <code>n</code> Upright (normal) • <code>it</code> Italic • <code>sl</code> Slanted (oblique) • <code>sc</code> Small caps • <code>ui</code> Upright italics • <code>ol</code> Outline The two last shapes are not available for most font families.
<code>\fontsize{size}{skip}</code>	Set font size. The first parameter is the font size to switch to; the second is the <code>\baselineskip</code> to use. The unit of both parameters defaults to pt. A rule of thumb is that the <code>\baselineskip</code> should be 1.2 times the font size.
<code>\selectfont</code>	The changes made by calling the four font commands described above do not come into effect until <code>\selectfont</code> is called.
<code>\usefont{enc}{family}{series}{shape}</code>	Equivalent to calling <code>\fontencoding</code> , <code>\fontfamily</code> , <code>\fontseries</code> and <code>\fontshape</code> with the given parameters, followed by <code>\selectfont</code> .

Table 1.11.: Low-level font commands

1.2.3. Parameters

The input file specification indicates the file to be formatted; T_EX uses `.tex` as a default file extension. If you omit the input file entirely, T_EX accepts input from the terminal. You specify command options by supplying a string as a parameter to the command; e.g.

```
latex \scrollmode\input foo.tex
```

will process `foo.tex` without pausing after every error.

Output files are always created in the current directory. When you fail to specify an input file name, T_EX bases the output names on the file specification associated with the logical name `TEX_OUTPUT`, typically `texput.log`.

1.3. L^AT_EX Packages

1.3.1. `amsmath`

`amsmath` consists of several packages to aid in typesetting math.

`\boldsymbol` Produces a bold math symbol. When not only a single symbol, but a complete equation has to be bold you might consider using the superior `\bm` command which takes better care of spacing.

`\pmb` Can be used for mathematic symbols, when no bold version is available. It prints the standard symbol several times slightly shifted and is inferior to a real bold font!

1.3.2. dcolumn

`\usepackage{dcolumn}`

Defines a system for defining columns of entries in an `array` or `tabular` which are to be aligned on a *decimal point*.

This package defines `D` to be a column specifier with three arguments:

`D{sep.tex}{sep.dvi}{decimal places}`

`sep.tex` should be a single character, this is used as the separator in the `.tex` file. It will usually be `'.'` or `','`.

`sep.dvi` is used as the separator in the output.

`decimal places` should be the maximum number of decimal places in the column.

`\newcolumntype` can be used to define own column types.

1.3.3. fontenc

`\usepackage[T1]{fontenc}`

chooses T1-fonts and allows correct automatic hyphenation.

1.3.4. german

Das `german` Paket muß geladen sein:

"a erzeugt ein *ä*
entsprechend "o, "u, "A, "O, "U, "s (ß)

"ck wird bei Trennung zu k-k

"ff wird bei Trennung zu ff-f
entsprechend "ll, "mm, "pp, "rr, "tt

`\glqq` oder "‘ erzeugt untere doppelte Anführungszeichen

`\grqq` oder "’ erzeugt obere doppelte Anführungszeichen

`\glq` erzeugt untere einfache Anführungszeichen

`\flqq` oder "< doppelte linke französische Anführungszeichen

`\frqq` oder "> doppelte rechte französische Anführungszeichen

`\flq` einfache linke französische Anführungszeichen

`\frq` einfache rechte französische Anführungszeichen

"| verhindert eine Ligatur an der Stelle

\- markiert mögliche Trennstelle, im restlichen Wort ist keine Trennung mehr möglich

"- Trennvorschlag, der Rest des Wortes kann noch automatisch getrennt werden

"" Trennvorschlag, bei dem bei einer Trennung kein Bindestrich eingefügt wird

"~ Bindestrich, an dem nicht getrennt werden darf

"= Bindestrich, an dem getrennt werden darf

`\dq` erzeugt ein nicht-typographisches gerades Anführungszeichen
`\texttt{\dq}`

`<` oder `\dq` ergibt das Zoll-Zeichen

1.3.5. graphics

Look up details in `grfguide`.

1.3.5.1. \includegraphics

`\includegraphics[] {picture-name}`

File extensions of the graphics should be omitted, so it's easier to change the format (for example for pdf or ps output).

1.3.5.2. \graphicspath

`\graphicspath{dir-list}`

`dir-list` is a list of absolute or relative paths, where the command `\includegraphics{}` searches for the graphic files:

`\graphicspath{{eps/}{tiff/}}` would cause the system to look in the subdirectories `eps` and `tiff` of the current directory. This is unix syntax, on a Mac it would be:

`\graphicspath{{:eps:}{:tiff:}}`

Under Windows/DOS use also the slash `/` to separate directories. Not all tex installations can handle filenames with spaces. So it's better to avoid them.

1.3.6. hyperref

For more infos read in the book *L^AT_EX Web Companion* or *Mit L^AT_EX ins Web*.

1.3.6.1. Options

4 use Acrobat 4 features (default: `false`).

a4paper, **a5paper**, **b5paper**, **legalpaper**, **letterpaper**, **executivepaper** use the specified paper size.

anchorcolor set color of anchors (default: `black`).

backref do bibliographical back references (default: `false`).

baseurl set base URL for document (default: `empty`).

bookmarks make bookmarks (default: `true`).

bookmarksnumbered put section numbers in bookmarks (default: `false`).

bookmarksopen open up bookmark tree (default: `false`).

bookmarksopenlevel level to which bookmarks are open (default: `\maxdimen`).

bookmarkstype to specify which 'toc' file to mimic toc.

breaklinks allow links to break over lines (default: `false`).

- citebordercolor** color of border around cites (default: 0 1 0).
- citecolor** color of citation links (default: green).
- colorlinks** color links (default: false).
- debug** provide details of anchors defined (same as verbose, default: false).
- draft** do not do any hyper linking (default: false).
- dvipdf, dvipdfm, dvips, dvipsone, dviwindo, hypertex, latex2html, pdftex, tex4ht, textures, ps2pdf, vtex** use the specified backend.
- extension** suffix of linked files (default: dvi).
- filebordercolor** color of border around file links (default: 0 .5 .5).
- filecolor** color of file links (default: cyan).
- frenchlinks** use small caps instead of color for links (default: false).
- hyperfigures** make figures hyper links (default: false).
- hyperindex** set up hyperlinked indices (default: true).
- hypertexnames** use guessable names for links (default: true).
- implicit** redefine L^AT_EX internals (default: true).
- linkbordercolor** color of border around links (default: 1 0 0).
- linkcolor** color of links (default: red).
- linktocpage** make page number, not text, be link on TOC, LOF and LOT (default: false).
- menubordercolor** color of border around menu links (default: 1 0 0).
- menucolor** color for menu links (default: red).
- naturalnames** use L^AT_EX-computed names for links (default: false).
- nesting** allow nesting of links (default: false).
- pageanchor** put an anchor on every page (default: true).
- pagebackref** backreference by page number (default: false).
- pagebordercolor** color of border around page links (default: 1 1 0).
- pagecolor** color of page links (default: red).
- pdfauthor** text for PDF Author field (default: empty).
- pdfborder** width of PDF link border (default: 0 0 1; 0 0 0 for colorlinks)
- pdfcenterwindow** position the document window center of the screen (default: false).
- pdfcreator** text for PDF Creator field (default: L^AT_EX with hyperref package).
- pdffitwindow** resize document window to fit document size (default: false).

pdfhighlight set highlighting of PDF links (default: `/I`).

pdfkeywords text for PDF Keywords field (default: empty).

pdfmenubar make PDF viewer's menu bar visible (default: `true`).

pdfnewwindow make links that open another PDF file start a new window (default: `false`).

pdfpagelayout set layout of PDF pages (default: empty).

pdfpagemode set default mode of PDF display (default: empty).

pdfpagescrop set crop size of PDF document (default: empty).

pdfpagetransition set PDF page transition style (default: empty).

pdfproducer text for PDF Producer field (default: empty).

pdfstartpage page at which PDF document opens (default: `1`).

pdfstartview starting view of PDF document (default: `/Fit`).

pdfsubject text for PDF Subject field (default: empty).

pdftitle text for PDF Title field (default: empty).

pdftoolbar make PDF toolbar visible (default: `true`).

pdfview PDF 'view' when on link traversal (default: empty).

pdfwindowui make PDF user interface elements visible (default: `true`).

plainpages do page number anchors as plain arabic (default: `true`).

raiselinks raise up links (for HyperTeX backend; default: `false`).

runbordercolor color of border around run links (default: `0 .7 .7`).

unicode Unicode encoded pdf strings (default: `true`).

urlbordercolor color of border around URL links (default: `0 1 1`).

urlcolor color of URL links (default: `magenta`).

verbose be chatty (default: `false`).

1.3.6.2. Commands

`\texorpdfstring{TeX-string}{pdf-string}` allows different texts for PDF or T_EX creation.
`\section{The role of \texorpdfstring{H$_2$O}{water}}`

`\href{URL}{Text}` the text will be resolved into a URL. The URL must be complete, according to a base-url which might be defined in a `\hyperbaseurl`.

`\href{http://www.karpfenteich.net/pit/}{Peter's input tips}`

`\hyperbaseurl{URL}` defines a base URL, which is preset to a given URL. This helps to make portable documents.

`\hyperimage{image-URL}` includes the picture of the URL reference.

`\hyperref{URL}{category}{name}{text}`, `\hyperref[mark]{text}` `text` is activated as a hyperlink to the point which is defined with a `\label` command with the symbolic name `mark`.

`\hyperlink{name}{text}`, `\hyertarget{name}{text}` `\hyperlink` creates an internal link to an object which is somewhere defined with a `\hypertarget` command.

For HTML the command `\hyperlink` inserts a `#` character in front of each link. Thus it'll refer to the topical document, while `\href` will expect a complete URL.

`\autoref{marker}` `\autoref` is a substitution for the standard `\ref` command. It inserts a context sensitive phrase.

`see \autoref{foo}`

for a `\label` in a section will output *see section 3* for an example.

There are macros to change the default output with the help of the `\renewcommand`:

<code>\figurename</code>	<code>*\figurename*</code>
<code>\tablename</code>	<code>*\tablename*</code>
<code>\partname</code>	<code>*\partname*</code>
<code>\appendixname</code>	<code>*\appendixname*</code>
<code>\equationname</code>	<code>*\equationname*</code>
<code>\Itemname</code>	<code>*\Itemname*</code>
<code>\chaptername</code>	<code>*\chaptername*</code>
<code>\sectionname</code>	<code>*\sectionname*</code>
<code>\subsectionname</code>	<code>*\subsectionname*</code>
<code>\subsubsectionname</code>	<code>*\subsubsectionname*</code>
<code>\paragraphname</code>	<code>*\paragraphname*</code>
<code>\Hfootnotename</code>	<code>*\Hfootnotename*</code>
<code>\AMSname</code>	<code>*\AMSname*</code>
<code>\theoremname</code>	<code>*\theoremname*</code>

`\nolinkurl` Allows line breaks in a verbatim like environment like `\url`, but without the hyperlink function. This is useful to define own commands like this example:

```
\newcommand*{\urlw}[1]{\href{http://www.#1}%
  {\nolinkurl{www.#1}}}
```

This new defined command `\urlw` allows the user to typeset a linkable (this function is offered by `\href`) `www.example.org` address in the latex source like this:

```
\urlw{example.org}
```

This will produce the string `www.example.org` which is hyperlinked and breakable (when the driver like pdf_TE_X allows line breaks in links).

1.3.7. inputenc

```
\usepackage[encoding name]{inputenc}
```

Allows to specify an input encoding for direct input of character codes > 127, e.g. accented characters.

Important encodings are:

`latin9`, `latin1` Linux, Unix, VMS

`ansinew` Windows (codepage 1252)

`cp850` OS/2, MSDOS (codpage 850)

`applemac` Apple Macintosh

1.3.8. KOMA-Script

The KOMA-Script package has a very detailed and well written documentation. Read the `scrguide` for further information! :-)

1.3.8.1. Page-Layout

```
\typearea[BCOR]{DIV}
\areaset[BCOR]{Breite}{Höhe}
```

1.3.8.2. Options

headinclude, headexclude, footinclude, footexclude will determine if headers and footers are used in the calculation of the page size. Default is exclude. Include will make the text area smaller.

DIV is a factor which determines how large the used page size is. This method takes care of good typography. Try values between 10 and 15. Higher values will cause smaller margins. An example would be: `DIV12`.

BCOR Binding correction. Takes care of the loss of the visible paper size when the paper is cut and a book is bound.

1.3.8.3. Letter Class `sclttr2`

This reference is based on the excellent `scrguien.pdf` from 2002-09-06.

Options

Defining Options Later

`\KOMAoptions{options list}` Use this to change options after loading the class. List single or multiple options, separated by commas.

Page Layout Options

`paper=format` Defines the paper format.

`BCOR=length, DIV=value, headlines=count` Divisor, binding correction, and number of headlines.

`enlargefirstpage` If *true* some more text would fit on the first page

Other Layout Options

`twoside` Activate possibilities of a double-sided document, but stay with with the one-sided layout as far as possible.

`cleardoublepage=style` Inserted pages by `\cleardoublepage` get one of these page styles:

- empty
- plain
- standard

`headsepline, footsepline` Insert a separator line below the head or above the foot.

`mpinclude, mpexclude` These two options of the `typearea` package should not be used with `sclttr2`.

pagenumber=position This option defines if and where a page number will be placed. Possible values are:

bot, foot Page number in foot, horizontal position not changed

botcenter, botcentered, botmiddle, footcenter, footcentered, footmiddle Page number in foot, centered

botleft, footleft Page number in foot, left justified

botright, footright Page number in foot, right justified

center, centered, middle Page number centered horizontally, vertical position not changed

false, no, off No page number

head, top Page number in head, horizontal position not changed

headcenter, headcentered, headmiddle, topcenter, topcentered, topmiddle Page number in head, centered

headleft, topleft Page number in head, left justified

headright, topright Page number in head, right justified

left Page number left, vertical position not changed

right Page number right, vertical position not changed
Default is **botcenter**

parskip=value Mark paragraphs alternatively by vertical skip. Possible values are:

false, off Paragraph indentation instead of vertical space

full, on, true One line vertical space between paragraphs; at least 1 em free space in the last line.

full* One line vertical space between paragraphs; at least a quarter of a line free space at the end of a paragraph.

full+ One line vertical space between paragraphs; at least a third of a line free space at the end of a paragraph.

full- One line vertical space between paragraphs; last line of a paragraph may be arbitrarily filled.

half Half a line vertical space between paragraphs; at least 1 em free space in the last line.

half* Half a line vertical space between paragraphs; at least a quarter of a line free space at the end of a paragraph.

half+ Half a line vertical space between paragraphs; at least a third of a line free space at the end of a paragraph.

half- One line vertical space between paragraphs; last line may be arbitrarily filled.
Default is **false**.

Font Options

fontsize=size Font size of the document. Defaults to 12pt.

Options for Letterhead and Address

fromalign Defines the placement of the from address in the letterhead. Possible values are:

center, centered, middle Return address centered.

false, no, off Standard design for return address.

left Left justified return address.

right Right justified return address.
Default is **left**

fromrule Allows to place a horizontal rule within return address. Possible values are:

afteraddress, below, true, yes Line below return address

aftername Line right below sender's name

false, no, off No line.
Default is **false**.

fromphone Defines whether the phone number will be part of the return address.

fromfax Defines whether the facsimile number will be part of the return address.

fromemail Defines whether the email address will be part of the return address.

fromurl Defines whether the URL will be part of the return address.

fromlogo Defines whether the logo will be part of the return address.

addrfield Defines whether a return address field will be set.

backaddress Defines whether a return address for window envelopes will be set.

subject Choose of your subject should have a title and if the subject should be set before or after the opening. Possible values are:

afteropening Set subject after opening.

beforeopening Set subject before opening.

titled Add title to subject.

untitled Do not add title to subject.
Defaults are **beforeopening** and **untitled**.

locfield Toggle between presets for the field width of additional sender attributes.

narrow Small sender supplement field.

wide Wide sender supplement field.
Default is **narrow**.

foldmarks Activates fold marks.

numericaldate Toggles between standard, language-dependent date presentation and a short, numerical one.

`refline` Defines that the reference line may extend beyond the normal type area.

Format Options

`draft` Toggles between the final and the draft version.

Letter Class Option Files

`\LoadLetterOption{name}` Load a `lco` file. `name` is the filename without suffix.
Predefined `lco` files:

`DIN` For A4 size paper, complying with DIN 676.

`DINmtext` For A4 size paper, complying with DIN 676, but using an alternate layout with more text on the first page.

`KOMAold` For A4 paper size using a layout close to the obsolete `scrletter` class.

`SN` For Swiss letters with address field on the right side, according to SN 010 130

`SNleft` For Swiss letters with address field on the left side.

`\LetterOptionNeedsPaperSize{option name}{paper size}` Useful to be able to emit warnings if the user selects a different paper size from the one the `lco` file is based on.

General Document Properties

Font Selection See also the section for *Changing Fonts* in KOMA-Script (still to be written).

`backaddress` Back address for a window envelope

`descriptionlabel` Label in a `description` environment

`fromaddress` Sender's address in letterhead

`fromname` Sender's address in letterhead if different from `fromaddress`

`pagefoot` Footer, sometimes header of a page

`pagehead` Header, sometimes footer of a page

`pagenumber` Page number in footer or header

`subject` Subject in the opening

`title` Headline in the opening

Page Style See also the section for *Page Style* in KOMA-Script (still to be written).

`empty` Entirely empty headers and footers

`plain` Empty head and only page number in header or footer

`headings` Automatic page headings

`myheadings` Manual page headings

Variables The main difference between a *command* and a *variable* is that a command usually triggers an action whereas a variable only consists of plain text. Furthermore a variable can have an additional description.

`\newkomavar[description]{name}`, `\newkomavar*[description]{name}`, `\addtoeffields{name}`

`\newkomavar` defines a new variable, addressed with `name`.

The command `\addtoeffields` adds the variable `name` to the reference fields. See section Business Line.

The command `\newkomavar*` works like `\newkomavar` with an additional call of the command `\addtoeffields`.

Overview of all variables:

`backaddress` Back address for window envelopes

`backaddressseparator` Separator within the back address

`ccseparator` Separator between title of additional recipients and additional recipients

`customer` Customer number

`date` Date

`emailseparator` Separator between e-mail name and e-mail address

`enclseparator` Separator between title of enclosure and and enclosures

`faxseparator` Separator between title of fax and fax number

`fromaddress` Sender's address without its name

`frombank` Sender's bank account

`fromemail` Sender's e-mail

`fromfax` Sender's fax number

`fromlogo` Commands for inserting the sender's logo

`fromname` Complete name of the sender

`fromphone` Sender's telephone number

`fromurl` URL of the sender

`invoice` Invoice number

`location` More details of the sender

`myref` Sender's reference

`place` Place

`placeseparator` Separator between place and date

`phoneseparator` Separator between title of telephone and telephone number

`signature` Signature beneath the ending of the letter

`specialmail` Special mail

`subject` Subject

`subjectseparator` Separator between title of subject and subject

`title` Letter title

`toname` Complete name of recipient

`toaddress` Address of recipient without its name

`yourmail` Date of recipient's mail

`yourref` Recipient's reference

`\setkomavar{name}[description]{content}`, `\setkomavar*{name}{description}` `\setkomavar` sets the content of the variable `name`. The optional argument sets the `description` of the variable. `\setkomavar*` only sets the `description`.

`\usekomavar[command]{name}`, `\usekomavar*[command]{name}` `\usekomavar` gives you access to the content of the variable `name`. `\usekomavar*` gives you access to the `description` of the variable `name`.

`\ifkomavareempty{name}{true}{false}`, `\ifkomavareempty*{name}{true}{false}` The `true` argument will be executed if the content or description is empty. Otherwise the `false` argument will be executed. The starred version handles the description of a variable, the variant without star the content.

Pseudo Lengths

`\@newplength{name}` Defines a new pseudo length

`\useplength{name}` Access value of pseudo length `name`

`\setlengthtoplength[factor]{length}{pseudo length}`, `\addtolengthlength[factor]{length}{pseudo length}`
Assign a multiple of `pseudo length` to a real `length`. Adds a multiple of `pseudo length` to `length`.

`\@setplength[factor]{pseudo length}{value}`, `\@addtoplength[factor]{pseudo length}{value}`
Assign a multiple of a `value` to a `pseudo length`. Adds `value` to `pseudo length`.

Letter Declaration

Letterhead

`firstheadvpos` Distance between the top of the sheet and the letterhead.

`firstheadwidth` Width of the letterhead.

`fromname`, `fromaddress`, `fromphone`, `fromfax`, `fromemail`, `fromurl`, `fromlogo` These variables give all statements concerning the sender necessary to create the letterhead.

`phoneseparator`, `faxseparator`, `emailseparator`, `urlseparator` Separators for variables in letterhead.

`\firsthead{construction}` Create letterhead freely.

Footer

`firstfootvpos` Distance between footer and upper border of the sheet.

`firstfootwidth` Width of letter's firstpage footer.

`\firstfoot{Construction}` Set definition for first page's footer

Address

`toaddrvpos`, `toaddrhpos` Distance between address window of a window envelope and the upper and left border of the sheet.

`toaddrwidth` Width of the address window.

`toaddrindent` Value of indentation for the address from the left border.

`backaddress`, `backaddressseparator`, `backaddrheight` Double backslashes within return address will be exchanged with `backaddressseparator`. Height of the return address field.

`specialmail`, `specialmailindent`, `specialmailrightindent` Optionally set between return address and addressee. `specialmailindent` and `specialmailrightindent` determine left and right indentation.

`toname`, `toaddress` Usually not set directly by the user.

`\begin{letter}[options]{address}` Major point of the letter class.

`\AtBeginLetter{command}` Hook for commands run at begin of a letter.

Sender's Extension

`locwidth` Declares width of sender's extensions.

`location` Content of sender's extension.

Business Line

`refvpos` Distance between upper border of sheet and business line.

`refwidth` Available width for the business line.

`refaftervskip` Vertical space to be inserted beneath business line.

`place`, `placeseparator` If all variables for business line are empty, the content of `place` and `placeseparator` will be set, followed by content of `date`.

`yourref`, `yourmail`, `myref`, `customer`, `invoice`, `date` Typical fields for business lines.

Title and Subject Line

`title` Additional tile. Set centered beneath the business line.

`subject`, `subjectseparator` Set the subject of the letter, optional prefixed by a label.

Further Issues

`tfoldmarkvpos`, `bfoldmarkvpos` Position of upper and lower fold mark.

`foldmarkhpos` Distance between fold marks and the sheet's left border.

`frombank` Internally not used yet.

`\nexthead{construction}`, `\nextfoot{construction}` Create letter head or foot for the following pages freely.

The Text

Opening

`\opening{opening}` Set the opening of the letter.

Footnotes See section *Footnotes* in KOMA-Script (still to be written).

Lists See section *Lists* in KOMA-Script (still to be written).

Margin Notes See section *Margin Notes* in KOMA-Script (still to be written).

Text Emphasis See section *Logical Markup of Text* in KOMA-Script (still to be written).

Closing Part

Closing

`signature` Includes an explanation for the inscription. Defaults to `\usekomavar{fromname}`.

`\closing{closing phrase}` Typesets the closing phrase followed by a vertical space and the contents of variable `signature`.

`sigindent`, `sigbeforevskip`, `\raggedsignature` Indentation of the box for closing phrase, inscription and signature (defaults to 0mm). Vertical space between closing phrase and signature (defaults to two lines). Alignment inside the box.

Postscript, Carbon Copy, and Enclosures

`\ps` Set a postscript.

`\cc{distribution list}`, `ccseparator` Set a distribution list. If the contents of `ccseparator` isn't empty then the name and the content of the variable is inserted prior to distribution list.

`\encl{enclosures}`, `enclseparator` Same structure as the distribution list.

Language Support

Language Selection

`\captionenglish`, `\captionUSenglish`, `\captionamerican`, `\captionbritish`, `\captionUKenglish`, `\captiongerman`
 If the used language selection scheme does not support switching the language the above commands can be used directly.

`\dateenglish`, `\dateUSenglish`, `\dateamerican`, `\datebritish`, `\dateUKenglish`, `\dategerman`, `\datengerman`, `\dat`
 Numerical date representation.

Language-Dependent Terms

`\yourrefname`, `\youremailname`, `\myrefname`, `\customername`, `\invoicename`, `\subjectname`, `\ccname`, `\enclname`, `\`
 These definitions can be modified to support a new language of for private customization.

Defining Language Terms

`\providcaptionname{language}{term}{definition}`, `\newcaptionname{language}{term}{defintion}`, `\renewcapti`
 Assign a *definition* for a *language* to a *term*.

Address Files and Circular Letters

`\adrentry{Lastname}{Firstname}{Address}{Telephone}{F1}{F2}{Comment}{Key}` Address entry for address files. The file extension has to be `.adr!`

`\addrentry{Lastname}{Firstname}{Address}{Telephone}{F1}{F2}{F3}{F4}{Key}` This command supports four freely definable parameters for an address entry.

`\adrchar{initial letter}`, `\addrchar{initial letter}` Separator for address entries. Ignored by `scrlettr2`.

1.3.8.4. Address Files with `scraddr`

`\InputAddressFile{file name}` Read the contents of the given address file. Filename extension has to be `.adr` and has to be omitted in the argument.

`\Name{Key}`, `\FirstName{Key}`, `\LastName{Key}`,
`\Address{Key}`, `\Telephone{Key}`, `\FreeI{Key}`,
`\FreeII{Key}`, `\Comment{Key}`, `\FreeIII{Key}`,
`\FreeIV{Key}` Access to the data in address file. `\Name` is a concatenation of `\FirstName` and `\LastName`.

`AdrFreeIVempty`, `AdrFreeIVshow`,
`AdrFreeIVwarn`, `AdrFreeIVstop` Package warning options. Defaults to `AdrFreeIVshow`.

1.3.9. `mathpazo`

Math fonts for the use with the Palatino font.

1.3.9.1. `\upDelta` / `\upOmega`

Math fonts for the use with the Palatino font.

Package options are:

[`slantedGreek`] Uppercase Greek will be typeset slanted.

[`noBBp1`] Do not use PazoMathBlackboardBold as (partial) blackboard bold font (e.g with `\mathbb{R}`).

[`osf`] Make the font family `pplj` (Palatino with old style numerals) the default roman font, and use an alternative version of the virtual math italic fonts (`zplmrj7m` and `zplmbj7m`) with upright Palatino old style numerals for use with the `\oldstylenums` command.

[`osfeqnum`] Use old style numerals for equation numbering.

1.3.10. `varioref`

Smarter version of the original latex2e cross reference commands. Generated strings are customizable, Babel options are recognized (further info in the package documentation).

`\vref`, `\vpageref` `\vref` is similar to `\ref` but adds an additional page reference, like *on the facing page* or *on page 27* whenever the corresponding `\label` is not on the same page.
`\vpageref` is a variation of `\pageref` with similar functionality.

`\vrefrange`, `\vpagerefrange` The `\v...range` commands take two labels as arguments and produce strings which depend on whether or not these labels fall onto a single page or on different pages.

`\vref*`, `\vpageref*`, `\vpagerefrange*` Star * variants do not add any space before the generated text for situations like:

```
(\vref{foo} ...)
```

1.3.11. Several Small Packages

Must be loaded with the usual `\usepackage` command. Further info for a package to be found in the documentation (dvi, pdf, ps) or in the `sty`-file itself.

1.3.11.1. `bm`

Bold math symbols or equations with better spacing than the `\boldsymbol` command. If possible load after other packages which redefine the fonts. When no bold font is available for a certain char `bm` will use *poor man's bold* `\pmb`.

`\bm` Produces bold math symbol or equation.

`\unboldmath` Can be used to set parts of an equation unbold.

1.3.11.2. `url`

Defines line breakable hyperlinked (uses `hyperref` package) verbatim input for urls and e-mail addresses.

Example: `\url{http://example.org}` outputs `http://example.org`.

Related commands are `\href` and `\nolinkurl`.

1.4. Generic Packages

1.4.1. P_lT_EX (tbd.)

tbd.

1.4.2. PSTricks

1.4.2.1. Overview

The idea behind the PSTricks package of Timothy van Zandt is to provide most of the PostScript language in L^AT_EX and T_EX syntax.

1.4.2.2. Commands and Parameters

Color

```
\newgray{color}{num}, \newrgbcolor{color}{num1 num2 num3}, \newhsbcolor{color}{num1 num2 num3}, \newcmykcolor{color}{num1 num2 num3 num4}
```

Note that these commands are obsolete for L^AT_EX. Load PSTricks with `\usepackage{pstcol}` and use the standard L^AT_EX color commands instead.

Setting graphics parameters

```
\psset{par1=value1,par2=value2,...}
```

Dimensions, coordinates and angles

```
\pssetlength{cmd}{dim}
```

```
unit=dim, xunit=dim, yunit=dim, yunit=dim Default: 1cm
```

```
\degrees[num]
```

```
\radians
```

Basic graphics parameters

```
linewidth=dim Default: .8pt
```

```
linecolor=color Default: black
```

```
showpoints=true/false Default: false
```

Lines and polygons

```
linearc=dim Default: 0pt
```

```
framearc=num Default: 0
```

```
cornersize=relative/absolute Default: relative
```

```
\psline*[par]{arrows}(x0,y0)(x1,y1)...(xn,yn)
```

```
\qline(coor0)(coor1)
```

```
\pspolygon*[par](x0,y0)(x1,y1)(x2,y2)...(xn,yn)
```

```
\psframe*[par](x0,y0)(x1,y1)
```

Arcs, circles and ellipses

```

\pscircle*[par] (x0,y0){radius}
\qdisk(coor){radius}
\pswedge*[par] (x0,y0){radius}{angle1}{angle2}
\psellipse*[par] (x0,y0) (x1,y1)
\psarc*[par]{arrows}(x,y){radius}{angleA}{angleB}
\psarcn*[par]{arrows}(x,y){radius}{angleA}{angleB}
arcsep=dim, arcsepA=dim, arcsepB=dim Default: 0pt

```

Curves

```

\psbezier*[par]{arrows}(x0,y0) (x1,y1) (x2,y2) (x3,y3)
\parabola*[par]{arrows}(x0,y0) (x1,y1)
\pscurve*[par]{arrows}(x1,y1) ... (xn,yn)
\psecurve*[par]{arrows}(x1,y1) ... (xn,yn)
\psccurve*[par]{arrows}(x1,y1) ... (xn,yn)
curvature=num1 num2 num3 Default: 1 .1 0

```

Dots

```

\psdots*[par] (x1,y1) (x2,y2) ... (xn,yn)
dotstyle=style Default: *
  Dots: *, o, +, triangle, triangle*, square, square*, pentagon, pentagon*, |
dotscale=num1 num2 Default: 1
dotangle=angle Default: 0

```

Grids

```

\psgrid(x0,y0) (x1,y1) (x2,y2)
gridwidth=dim Default: .8pt
gridcolor=color Default: black
griddots=num Default: 0
gridlabels=dim Default: 10pt
gridlabelcolor=color Default: black
subgriddiv=int Default: 5

```

subgridwidth=dim Default: .4pt

subgridcolor=color Default: gray

subgriddots=num Default: 0

Plots

\fileplot*[par]{file}

\dataplot*[par]{commands}

\savedata{command}[data]

\readdata{command}{file}

\listplot*[par]{list}

\psplot*[par]{xmin}{xmax}{function}

\parametricplot*[par]{tmin}{tmax}{function}

plotstyle=style Default: line

plotpoints=int Default: 50

Coordinate systems

origin={coor} Default: 0pt,0pt

swapaxes=true Default: false

Line styles

linestyle=style Default: solid

dash=dim1 dim2 Default: 5pt 3pt

dotsep=dim Default: 3pt

border=dim Default: 0pt

bordercolor=color Default: white

doubleline=true/false Default: false

doublesep=dim Default: 1.25\pslinewidth

doublecolor=color Default: white

shadow=true/false Default: false

shadowsize=dim Default: 3pt

shadowangle=angle Default: -45

shadowcolor=color Default: darkgray

dimen=outer/inner/middle Default: outer

Fill styles

fillstyle=style Default: none

fillcolor=color Default: white

hatchwidth=dim Default: .8pt

hatchsep=dim Default: 4pt

hatchcolor=color Default: black

hatchangle=rot Default: 45

Arrowheads and such

arrows=style Default: -

Arrows: -, <->, >-<, <<->>, >>-<<, |-, |*|*, [-], (-), o-o, ***, oo-oo, ***-**, c-c, cc-cc, C-C

arrowsize=dim num Default: 2pt 3

arrowlength=num Default: 1.4

arrowinset=num Default: .4

tbar size=dim num Default: 2pt 5

bracketlength=num, rbracketlength=num Default: .15

dotsize=dim num Default: .5pt 2.5

arrowscale=num1 num2 Default: 1

Custom styles

`\newsobject{name}{object}{par1=value1,...}`

`\newsstyle{name}{par1=value1,...}`

The basics

`\pscustom*[par]{commands}`

Parameters

linetype=int Default: 0

Graphics objects

liftpen=0/1/2 Default: 0

Safe tricks

`\newpath`

`\moveto(coor)`

`\closepath`

`\stroke[par]`

`\fill[par]`

`\gsave`

`\grestore`

`\translate(coor)`

`\scale{num1 num2}`

`\rotate{angle}`

`\swapaxes`

`\msave`

`\mrestore`

`\openshadow[par]`

`\closedshadow[par]`

`\movepath(coor)`

Pretty safe tricks

`\lineto(coor)`

`\rlineto(coor)`

`\curveto(x1,y1)(x2,y2)(x3,y3)`

`\rcurveto(x1,y1)(x2,y2)(x3,y3)`

For hackers only

`\code{code}`

`\dim{dim}`

`\coor(x1,y1)(x2,y2) ... (xn,yn)`

`\rcoor(x1,y1)(x2,y2) ... (xn,yn)`

`\file{file}`

`\arrows{arrows}`

`\setcolor{color}`

Pictures

`\pspicture*[baseline](x0,y0)(x1,y1)`

`\endpspicture`

Placing and rotating whatever

`\rput*[refpoint]{rotation}(x,y){stuff}`

`\uput*[labelsep][refangle]{rotation}(x,y){stuff}`

`\pslabelsep`

labelsep=dim Default: 5pt

Repetition

`\multirput*[refpoint]{angle}(x0,y0)(x1,y1){int}{stuff}`

`\multips{angle}(x0,y0)(x1,y1){int}{graphics}`

Axes

		<code>\psaxes*[par]{arrows}(x0,y0)(x1,y1)(x2,y2)</code>	
<i>Horizontal</i>	<i>Vertical</i>	<i>Dflt</i>	<i>Description</i>
Ox=num	Oy=num	0	Label at origin.
Dx=num	Dy=num	1	Label increment.
dx=dim	oy=dim	0pt	Dist btwn labels.

labels=all/x/y/none Default: all

showorigin=true/false Default: true

ticks=all/x/y/none Default: all

tickstyle=full/top/bottom Default: full

ticksize=dim Default: 3pt

`\psxlabel`, `\psylabel`

axesstyle=axes/frame/none Default: axes

Framed boxes

framesep=dim Default: 3pt

boxsep=true/false Default: true

```

\psframebox*[par]{stuff}
\psdblframebox*[par]{stuff}
\psshadowbox*[par]{stuff}
\pscirclebox*[par]{stuff}
\cput*[par]{angle}(x,y){stuff}
\psovalbox*[par]{stuff}

```

Clipping

```

\clipbox[dim]{stuff}
\psclip{graphics} ... \endpsclip

```

Rotation and scaling boxes

```

\rotateleft{stuff}
\rotateright{stuff}
\rotatedown{stuff}
\scalebox{num1 num2}{stuff}
\scaleboxto(x,y){stuff}

```

Nodes

```

\rnode[refpoint]{name}{stuff}
\Rnode(x,y){name}{stuff}
\RnodeRef
\pnode(x,y){name}
\cnode*[par](x,y){radius}{name}
\circlenode*[par]{name}{stuff}
\cnodeput*[par]{angle}(x,y){name}{stuff}
\ovalnode*[par]{name}{stuff}

```

Node connections

```

nodesep=dim Default: 0
offset=dim Default: 0
arm=dim Default: 10pt

```

```

angle=angle Default: 0
arcangle=angle Default: 8
ncurv=num Default: .67
loopsize=dim Default: 1cm
\ncline*[par]{arrows}{nodeA}{nodeB}
\ncLine*[par]{arrows}{nodeA}{nodeB}
\nccurve*[par]{arrows}{nodeA}{nodeB}
\ncarc*[par]{arrows}{nodeA}{nodeB}
\ncbar*[par]{arrows}{nodeA}{nodeB}
\ncdiag*[par]{arrows}{nodeA}{nodeB}
\ncdiagg*[par]{arrows}{nodeA}{nodeB}
\ncangle*[par]{arrows}{nodeA}{nodeB}
\ncangles*[par]{arrows}{nodeA}{nodeB}
\ncloop*[par]{arrows}{nodeA}{nodeB}
\nccircle*[par]{arrows}{node}{radius}
\pcline*[par]{arrows}(x1,y1)(x2,y2)
\pccurve*[par]{arrows}(x1,y1)(x2,y2)
\pcarc*[par]{arrows}(x1,y1)(x2,y2)
\pcbar*[par]{arrows}(x1,y1)(x2,y2)
\pcdiag*[par]{arrows}(x1,y1)(x2,y2)
\pcangle*[par]{arrows}(x1,y1)(x2,y2)
\pcloop*[par]{arrows}(x1,y1)(x2,y2)

```

Attaching labels to node connections

```

\lput*[refpoint]{rotation}(pos){stuff}
\aput*[labelsep]{angle}(pos){stuff}
\bput*[labelsep]{angle}(pos){stuff}
\mput*[refpoint]{stuff}
\Aput*[labelsep]{stuff}
\Bput*[labelsep]{stuff}

```

Coils and zigzags

```

\pscoil*[par]{arrows}(x0,y0)(x1,y1)
\psCoil*[par]{angle1}{angle2}
\pszizzag*[par]{arrows}(x0,y0)(x1,y1)
coilwidth=dim Default: 1cm
coilheight=num Default: 1
coilarm=dim Default: .5cm
coilaspect=angle Default: 45
coilinc=angle Default: 10
\nccoil*[par]{arrows}{nodeA}{nodeB}
\nczigzag*[par]{arrows}{nodeA}{nodeB}
\pccoil*[par]{arrows}(x1,y1)(x2,y2)
\pczigzag*[par]{arrows}(x1,y1)(x2,y2)

```

Special coordinates `\SpecialCoor`

<i>Coordinate</i>	<i>Example</i>	<i>Description</i>
(x,y)	(3,4)	Cartesian coordinate.
(r;a)	(3;110)	Polar coordinate.
(node)	(A)	Center of node.
([par]node)	([angle=45]A)	Relative to node.
(!ps)	(!5 3.3 2 exp)	Raw PostScript.
(coor1 coor2)	(A lin;30)	Combination.

<i>Angle</i>	<i>Example</i>	<i>Description</i>
num	45	Angle.
(coor)	(-1,1)	Coordinate (vector).
!ps	!33 sqrt	Raw PostScript.

Table 1.12.: Special coordinates and angles

```
\NormalCoor
```

Overlays

```

\overlaybox stuff\endoverlaybox
\psoverlay{string}
\putoverlaybox{string}
gradbegin=color Default: gradbegin
gradend=color Default: gradend
gradlines=int Default: 500

```

gradmidpoint=num Default: .9

gradangle=angle Default: 0

Typesetting text along a path

`\pstextpath[pos](x,y){graphics object}{text}`

Stroking and filling character paths

`\pscharpath*[par]{text}`

`\pscharclip*[par]{text} ... \endpscharclip`

Exporting EPS files

`\TeXtoEPS ... \endTeXtoEPS`

`\PSTtoEPS[par]{file}{graphics objects}`

bbllx=dim Default: -1pt

bbly=dim Default: -1pt

bburx=dim Default: 1pt

bbury=dim Default: 1pt

headerfile=file Default: s

headers=none/all/user Default: none

Boxes

`\psmathboxtrue, \psmathboxfalse`

`\everypsbox{commands}`

`\pslongbox{name}{cmd}`

`\psverbboxtrue, \psverbboxfalse`

Tips and More Tricks

`\pslbrace, \psrbrace`

1.4.2.3. List of Additional PSTricks Packages

psgo <<ftp://ftp.dante.de/pub/tex/graphics/pstricks/contrib/psgo/>> Draw Go diagrams

pst-blur <<ftp://ftp.dante.de/pub/tex/graphics/pstricks/contrib/pst-blur/>> PSTricks package for "blurred" shadows

pst-euc <<http://dominique.rodriiguez.9online.fr/pst-eucl/>> Géométrie en L^AT_EX et PSTricks

pst-fr3d <<http://www.tug.org/applications/PSTricks/More/pst-fr3d.tar.gz>> Three dimensional framed Boxes

pst-ghsb <<ftp://ftp.dante.de/pub/tex/graphics/pstricks/contrib/pst-ghsb/>> PSTricks package for HSB Gradients

pst-gr3d <<ftp://ftp.dante.de/pub/tex/graphics/pstricks/contrib/pst-gr3d/>> PSTricks package for 3D grids

pst-lens <<http://members.aol.com/ManuelLuque2/optique.htm>> Optique géométrique

pst-node <<ftp://ftp.dante.de/tex-archive/graphics/pstricks/generic/>> PSTricks package for nodes

pst-ob3d <<http://members.aol.com/Mluque5130/pst-ob3d.tar.gz>> A PSTricks package for three dimensional basic objects

pst-osci <<http://www.tug.org/applications/PSTricks/Physics/Oscilloscopes/>> Oscilloscopes with PSTricks

pst-poly <<ftp://ftp.dante.de/pub/tex/graphics/pstricks/contrib/pst-poly/>> Polygons with PSTricks

pst-tree <<ftp://ftp.dante.de/tex-archive/graphics/pstricks/generic/>> PSTricks package for trees

pst-uml <<http://www.ensta.fr/~diam/latex/pst-uml/>> Draw easily diagrams with UML notation

vaucansom.sty <<http://perso.enst.fr/~lombardy/Vauc/>> Drawing automata

1.5. fontinst (tbd.)

tbd.

1.6. ConTeXt (tbd.)

1.6.1. Overview

ConTeXt is based on PlainT_EX and MetaPost and gets controlled with some Perl scripts. It's mainly targeted towards layout oriented users and especially useful for presentations. It's extensible by the use of modules and has in it's base distribution already a large amount of functionality. The preferred output format is PDF, but DVI is possible too.

Some of the unique features of ConTeXt are:

- Inline XML (including MathML, ChemML, and PhysML)
- really good XML support in general
- inline MetaPost

1.7. Texinfo (tbd.)

tbd.

2. pdfTEX

NOTE

The contents of this section was taken from `pdftex(1)`.

2.1. Options

pdfTEX understands the following command line options.

2.1.1. `--file-line-error-style`

Print error messages in the form `file:line:error` which is similar to the way many compilers format them.

2.1.2. `--fmt format`

Use `format` as the name of the format to be used, instead of the name by which pdfTEX was called or a `%&` line.

2.1.3. `--help`

Print help message and exit.

2.1.4. `--ini`

Be `pdfnitetex`, for dumping formats; this is implicitly true if the program is called as `pdfnitetex`.

2.1.5. `--interaction mode`

Sets the interaction mode. The mode can be one of `batchmode`, `nonstopmode`, `scrollmode`, and `errorstopmode`. The meaning of these modes is the same as that of the corresponding `\commands`.

2.1.6. `--ipc`

Send DVI or PDF output to a socket as well as the usual output file. Whether this option is available is the choice of the installer.

2.1.7. `--ipc-start`

As `--ipc`, and starts the server at the other end as well. Whether this option is available is the choice of the installer.

2.1.8. `--jobname name`

Use `name` for the job name, instead of deriving it from the name of the input file.

2.1.9. --kpathsea-debug bitmask

Sets path searching debugging flags according to the `bitmask`. See the Kpathsea manual for details.

2.1.10. --maketex fmt

Enable `mktexfmt`, where `fmt` must be one of `tex` or `tfm`.

2.1.11. --no-maketex fmt

Disable `mktexfmt`, where `fmt` must be one of `tex` or `tfm`.

2.1.12. --output-comment string

Use `string` for the DVI file comment instead of the date.

2.1.13. --parse-first-line

If the first line of the main input file begins with `%&` parse it to look for a dump name or a `--translate-file` option.

2.1.14. --progrname name

Pretend to be program `name`. This affects both the format used and the search paths.

2.1.15. --recorder

Enable the filename recorder. This leaves a trace of the files opened for input and output in a file with extension `.fls`.

2.1.16. --shell-escape

Enable the `\write18{command}` construct. The `command` can be any Bourne shell command. This construct is normally disallowed for security reasons.

2.1.17. --translate-file tcxname

Use the `tcxname` translation table.

2.1.18. --version

Print version information and exit.

2.2. Environment

See the Kpathsearch library documentation (the ‘Path specifications’ node) for the details of how the environment variables are used when searching. The `kpsewhich` utility can be used to query the values of the variables.

One caveat: In most pdfTeX formats, you cannot use `~` in a filename you give directly to pdfTeX, because `~` is an active character, and hence is expanded, not taken as part of the filename. Other programs, such as Metafont, do not have this problem.

TEXMFOUTPUT Normally, pdfTEX puts its output files in the current directory. If any output file cannot be opened there, it tries to open it in the directory specified in the environment variable **TEXMFOUTPUT**. There is no default value for that variable. For example, if you say **pdftex paper** and the current directory is not writable, if **TEXMFOUTPUT** has the value **/tmp**, pdfTEX attempts to create **/tmp/paper.log** (and **/tmp/paper.pdf**, if any output is produced.)

TEXINPUTS Search path for **\input** and **\openin** files. This should probably start with **“.”**, so that user files are found before system files. An empty path component will be replaced with the paths defined in the **texmf.cnf** file. For example, set **TEXINPUTS** to **“./home/usr/tex:”** to prepend the current directory and **“/home/user/tex”** to the standard search path.

TEXFONTS Search path for font metric (**.tfm**) files.

TEXFORMATS Search path for format files.

TEXPOOL Search path for **pdfninitex** internal strings.

TEXEDIT Command template for switching to editor. The default, usually **vi**, is set when pdfTEX is compiled.

3. Metafont (tbd.)

tbd.

4. MetaPost

NOTE

The contents of this section was taken from `mpost(1)`.

MetaPost is a programming language much like Knuth's Metafont except that it outputs PostScript programs instead of bitmaps. Borrowed from Metafont are the basic tools for creating and manipulating pictures. These include numbers, coordinate pairs, cubic splines, affine transformations, text strings, and boolean quantities. Additional features facilitate integrating text and graphics and accessing special features of PostScript such as clipping, shading, and dashed lines. Another feature borrowed from Metafont is the ability to solve linear equations that are given implicitly, thus allowing many programs to be written in a largely declarative style. By building complex operations from simpler ones, MetaPost achieves both power and exibility.

4.1. Options

MetaPost understands the following command line options.

4.1.1. `--file-line-error-style`

Print error messages in the form `file:line:error` which is similar to the way many compilers format them.

4.1.2. `--help`

Print help message and exit.

4.1.3. `--ini`

Be **inimpost**, for dumping bases; this is implicitly true if the program is called as **inimpost**.

4.1.4. `--interaction mode`

Sets the interaction mode. The mode can be one of `batchmode`, `nonstopmode`, `scrollmode`, and `errorstopmode`. The meaning of these modes is the same as that of the corresponding commands.

4.1.5. `--jobname name`

Use `name` for the job name, instead of deriving it from the name of the input file.

4.1.6. `--kpathsea-debug bitmask`

Sets path searching debugging flags according to the `bitmask`. See the Kpathsea manual for details.

4.1.7. `--mem mem`

Use `mem` as the name of the mem to be used, instead of the name by which MetaPost was called or a `%&` line.

4.1.8. --parse-first-line

If the first line of the main input file begins with `%&` parse it to look for a dump name or a `--translate-file` option.

4.1.9. --progrname name

Pretend to be program `name`. This affects both the format used and the search paths.

4.1.10. --recorder

Enable the filename recorder. This leaves a trace of the files opened for input and output in a file with extension `.fls`.

4.1.11. -T

Produce TROFF output.

4.1.12. --translate-file tcxname

Use the `tcxname` translation table.

4.1.13. --troff

As -T.

4.1.14. --version

Print version information and exit.

4.2. Environment

See the Kpathsearch library documentation (the ‘Path specifications’ node) for the details of how the environment variables are use when searching. The `kpsewhich` utility can be used to query the values of the variables.

If the environment variable `TEXMFOUTPUT` is set, MetaPost attempts to put its output files in it, if they cannot be put in the current directory.

Here is a list of the environment variables which affect the behavior of MetaPost:

`MPINPUTS` Search path for input files.

`MFINPUTS` Auxiliary search path for input files with `.mf` extensions.

`MPSUPPORT` Directory for various tables for handling included `TEX` and troff.

`MPXCOMMAND` The name of a shell script that converts embedded typesetting commands to a form that MetaPost understands. Defaults: `makempx` for `TEX` and `troffmpx` for troff.

`TEX` The version of `TEX` - or `LATEX` - to use when processing `btex` and `verbatimtex` commands. Default `TEX`. This version of MetaPost allows you to use a ‘`%&format`’ line instead.

TROFF The troff pipeline for **btex** and **verbatimtex** commands. Default `eqn -d\$\$ | troff`

MPEDIT A command template for invoking an editor.

A `.mem` file is a binary file that permits fast loading of macro packages. MetaPost reads the default `plain.mem` unless another `.mem` file is specified at the start of the first line with an `&` just before it. There is also an `mfplain.mem` that simulates plain Metafont so that MetaPost can read `.mf` fonts. (Plain Metafont is described in *The Metafontbook*).

Experts can create `.mem` files by invoking **inimpost** and giving macro definitions followed by a `dump` command.

The MetaPost language is similar to Metafont, but the manual *A User's Manual for MetaPost* assumes no knowledge of Metafont. MetaPost does not have bitmap output commands or Metafont's online display mechanism.

5. BibT_EX

NOTE

The contents of this section was taken from the HTML helppages for BibT_EX of Norman Walsh (Version 1.0, 12 Apr 94).

Invokes the BibT_EX utility to compile a bibliography file for L^AT_EX. Full details can be found in "L^AT_EX: A Document Preparation System" by Leslie Lamport.

5.1. Parameters

`bibliography-file-spec`

Specifies the name of the bibliography database file to be compiled by BibT_EX. If the file specification does not include a file type, BibT_EX assumes a default type of BIB.

5.2. Command Qualifiers

`/BIBINPUTS /BIBINPUTS=(name,...)`

Specify directories containing input files, and the order in which they will be searched to locate each input file. A null value in the list indicates the current directory. The search procedure T_EX uses to locate input files is to first search your default directory and then search each of the directories specified by the `/BIBINPUTS` option.

Default is `/BIBINPUTS=(TEX_BIB:)`; T_EX looks in the directory associated with the logical name `TEX_BIB`.
`/STATS /STATS /NOSTATS [D]`

This qualifier is used while debugging `.BST` files to determine BibT_EX memory usage. `/TEXINPUTS /TEXINPUTS=(name,...)`

Specify directories containing input files, and the order in which they will be searched to locate each input file. A null value in the list indicates the current directory. The search procedure T_EX uses to locate input files is to first search your default directory and then search each of the directories specified by the `/TEXINPUTS` option.

Default is `/TEXINPUTS=(TEX_INPUTS)`; T_EX looks in the directory associated with the logical name `TEX_INPUTS`. `/TRACE /TRACE /NOTRACE [D]`

This qualifier is used while debugging `.BST` files to follow program flow.

5.3. bib files

This help entry contains the same information as Appendix B of the L^AT_EX manual. It describes the format of a bibliography database (`.BIB`) file.

A bibliography database file may contains two types of entry - an abbreviation definition or a reference entry for citation.

5.3.1. @STRING command

The `@STRING` command is used to define abbreviations for use by BibT_EX within the bibliography database file. The command

```
@string{jgg1 = "Journal of Gnats and Gnus, Series~1"}
```

defines 'jgg1' to be the abbreviation for the string "Journal of Gnats and Gnus, Series~1". Parentheses can be used in place of the outermost braces in the @string command, and braces can be used instead of the quotation marks. The text must have matching braces.

The case of letters is ignored in an abbreviation as well as in the command name @string, so the command above could have been written:

```
@STRING{JgG1 = "Journal of Gnats and Gnus, Series~1"}
```

A @string command can appear anywhere before or between entries in a bibliography database file. However, it must come before any use of the abbreviation, so a sensible place for @string commands is at the beginning of the file. A @string command in the bibliography database file takes precedence over a definition made by the bibliography style, so it can be used to change the definition of an abbreviation such as 'Feb'.

5.3.2. Entry Format

A bibliography database file contains a series of reference entries like the following:

```
@BOOK{kn:gnus,
  AUTHOR = "Donald E. Knudson",
  TITLE  = "1966 World Gnus Almanac",
  PUBLISHER = {Permafrost Press},
  ADDRESS = {Novisibirsk}      }
```

The @BOOK states that this is an entry of type book. various entry types are described below. The 'kn:gnus' is the citation key, as it appears in the argument of a \cite command referring to the entry.

This entry has four fields, named AUTHOR, TITLE, PUBLISHER and ADDRESS. The meanings of these and other fields are described below. A field consists of the name, an '=' character with optional space around it, followed by its text. The text of a field is a string of characters, with no unmatched braces, surrounded by either a pair of braces or a pair of ''' characters. Entry fields are separated from one another, and from the citation key, by commas. A comma may have optional space around it.

The outermost braces that surround the entire entry may be replaced by parentheses. As in T_EX input files, an end-of-line character counts as a space and one space is equivalent to many spaces. Unlike T_EX, BIBTEX ignores the case of letters in the entry type, citation key and field names. The above entry could have been typed as follows:

```
@BOOK(kn:gnus, author = {Donald E. Knudson},
  TITLE  = "1966 World Gnus Almanac",
  PUBLISHER = {Permafrost Press},
  ADDRESS = {Novisibirsk}      )
```

However, the case of letters does matter to L^AT_EX, so the citation key ("kn:gnus" in the example above) should appear exactly the same in all \cite commands in the L^AT_EX input file.

The quotes or braces can be omitted around text consisting entirely of numerals. The following two fields are equivalent:

```
Volume = "27"           Volume = 27
```

5.3.3. Entry Types

When entering a reference in the bibliography database, the first thing to decide is what type of entry it is. No fixed classification scheme can be complete, but BIBTEX provides enough entry types to handle almost any reference reasonably well.

References to different types of publications contain different information; a reference to a journal might include the volume and number of the journal, which is usually not meaningful for a book. Therefore, database entries of different types have different fields for each entry type, the fields are divided into three classes:

Required omitting the field will produce an error message and may result in a badly formatted bibliography entry. If the required information is not meaningful, you are using the wrong entry type.

Optional the field's information will be used if present, but can be omitted without causing any formatting problems. A reference should contain any available information that might help the reader, so you should include the optional field if it is applicable.

Ignored the field is ignored. `BIBTEX` ignores any field that is not required or optional, so you can include any fields you want in a bibliography entry. It's often a good idea to put all relevant information about a reference in its bibliography entry - even information that may never appear in the bibliography. For example, if you want to keep an abstract of a paper in a computer file, put it in an 'abstract' field in the paper's bibliography entry. The bibliography database file is likely to be as good a place as any for the abstract, and it is possible to design a bibliography style for printing selected abstracts.

`BIBTEX` ignores the case of letters in the entry type.

5.3.3.1. article entry

An article from a journal or magazine.

Format:

```
@ARTICLE{citation_key,
          required_fields [, optional_fields] }
```

Required fields: author, title, journal, year

Optional fields: volume, number, pages, month, note, key

5.3.3.2. book entry

A book with an explicit publisher.

Format:

```
@BOOK{citation_key,
       required_fields [, optional_fields] }
```

Required fields: author or editor, title, publisher, year

Optional fields: volume, series, address, edition, month, note, key

5.3.3.3. booklet entry

A work that is printed and bound, but without a named publisher or sponsoring institution.

Format:

```
@BOOKLET{citation_key,
          required_fields [, optional_fields] }
```

Required fields: title

Optional fields: author, howpublished, address, month, year, note, key

5.3.3.4. conference entry

An article in the proceedings of a conference. This entry is identical to the 'inproceedings' entry and is included for compatibility with another text formatting system.

Format:

```
@CONFERENCE{citation_key,
             required_fields [, optional_fields] }
```

Required fields: author, title, booktitle, year

Optional fields: editor, pages, organization, publisher, address, month, note, key

5.3.3.5. inbook entry

A part of a book, which may be a chapter and/or a range of pages.

Format:

```
@INBOOK{citation_key,
         required_fields [, optional_fields] }
```

Required fields: author or editor, title, chapter and/or pages, publisher, year

Optional fields: volume, series, address, edition, month, note, key

5.3.3.6. incollection entry

A part of a book with its own title.

Format:

```
@INCOLLECTION{citation_key,
              required_fields [, optional_fields] }
```

Required fields: author, title, booktitle, year

Optional fields: editor, pages, organization, publisher, address, month, note, key

5.3.3.7. inproceedings entry

An article in the proceedings of a conference.

Format:

```
@INPROCEEDINGS{citation_key,
                required_fields [, optional_fields] }
```

Required fields: author, title, booktitle, year

Optional fields: editor, pages, organization, publisher, address, month, note, key

5.3.3.8. manual entry

Technical documentation.

Format:

```
@MANUAL{citation_key,
         required_fields [, optional_fields] }
```

Required fields: title

Optional fields: author, organization, address, edition, month, year, note, key

5.3.3.9. mastersthesis entry

A Master's thesis.

Format:

```
@MASTERSTHESIS{citation_key,
                required_fields [, optional_fields] }
```

Required fields: author, title, school, year

Optional fields: address, month, note, key

5.3.3.10. misc entry

Use this type when nothing else seems appropriate.

Format:

```
@MISC{citation_key,
      required_fields [, optional_fields] }
```

Required fields: none

Optional fields: author, title, howpublished, month, year, note, key

5.3.3.11. phdthesis entry

A PhD thesis.

Format:

```
@PHDTHESIS{citation_key,
            required_fields [, optional_fields] }
```

Required fields: author, title, school, year

Optional fields: address, month, note, key

5.3.3.12. proceedings entry

The proceedings of a conference.

Format:

```
@PROCEEDINGS{citation_key,
              required_fields [, optional_fields] }
```

Required fields: title, year

Optional fields: editor, publisher, organization, address, month, note, key

5.3.3.13. techreport entry

A report published by a school or other institution, usually numbered within a series.

Format:

```
@TECHREPORT{citation_key,
            required_fields [, optional_fields] }
```

Required fields: author, title, institution, year

Optional fields: type, number, address, month, note, key

5.3.3.14. unpublished entry

A document with an author and title, but not formally published.

Format:

```
@UNPUBLISHED{citation_key,
              required_fields [, optional_fields] }
```

Required fields: author, title, note

Optional fields: month, year, key

5.3.4. Field Text

The text of the field is enclosed in braces or double quote characters. A part of the text is said to be enclosed in braces if it lies inside a matching pair of braces other than the ones enclosing the entire entry or the entire field text.

BIBTEX manipulates the case of letters in the field text as described in the subtopics below.

5.3.4.1. Names

The text of an author or editor field represents a list of names. The bibliography style determines the format in which the name is printed: whether the first name or last name appears first, if the full first name or just the first initial is used, etc. The bibliography file entry simply tells BIBTEX what the name is.

You should type an author's complete name and let the bibliography style decide what to abbreviate. (But an author's complete name may be "Donald~E. Knuth" or even "J.~P.~Morgan"; you should type it the way the author would like it to appear, if that's known.)

Most names can be entered in the obvious way, either with or without a comma, as in the following examples:

"John Paul Jones" "Jones, John Paul" "Ludwig von Beethoven" "von Beethoven, Ludwig"

Some people have multiple last names - for example, Per Brinch Hansen's last name is Brinch~Hansen. His name should be typed with a comma:

"Brinch Hansen, Per"

To understand why, you must understand how BIBTEX handles names (for what follows, a "name" corresponds to a person).

Each name consists of four parts: First, von, Last, and~Jr; each part consists of a (possibly empty) list of name-tokens. For example the First part of Per Brinch~Hansen's name has the single token "Per"; the Last part has two, "Brinch" and "Hansen"; and the von and Jr parts are empty. If you had typed

"Per Brinch Hansen"

instead, BIBTEX would erroneously think "Brinch" were a First-part token, just as "Paul" is a First-part token in "John~Paul Jones".

Here's another example:

"Charles Louis Xavier Joseph de la Vallee Poussin"

This name has four tokens in the First part, two in the von, and two in the Last. Here BIBTEX knows where one part ends and the other begins because the tokens in the von part begin with lower-case letters.

If you want BIBTEX to consider something a single token, enclose it in braces. You should do this, for example, if a comma is part of a name:

"{Barnes and Noble, Inc.}" "{Barnes and} {Noble, Inc.}" "{Barnes} {and} {Noble,} {Inc.}"

The braces surrounding the comma keep "Inc." from being interpreted as a First token; this name has only a Last part, with either one, two, or four tokens (there must be a space separating the tokens in the second and third forms). Probably the second form is slightly more meaningful, but don't lose sleep over this since only rarely will an institution be an author or editor.

So the two names

"von Beethoven, Ludwig" "{von Beethoven}, Ludwig"

are considered by BIBTEX to be different names. In the first, "Beethoven" is the Last part and "von" is the von part; in the second, which in this case happens to be incorrect, the Last part has a single token and there's no von part. The bibliography style will probably print both the same, but it may alphabetize and label them differently.

"Juniors" pose a special problem. Most people having "Jr." in their name precede it with a comma. Such a name should be entered as follows:

"Ford, Jr., Henry"

However, a few people do not use a comma. They are handled by considering the "Jr." to be part of the last Last token:

```
"{Steele Jr.}, Guy L." "Guy L. {Steele Jr.}"
```

This name has no Jr part.

To summarize, you may type a name in one of three forms:

```
"First von Last" "von Last, First" "von Last, Jr, First"
```

You may almost always use the first form; you shouldn't if either there's a Jr part or the Last part has multiple tokens but there's no von part.

If there are multiple authors or editors, their names must be separated by the word "and", surrounded by spaces, not enclosed in braces:

```
AUTHOR = "Ralph Alpher and Bethe, Hans and George Gamow"
```

Since `BIBTEX` interprets commas as separating parts of a name and "and" as separating names themselves, this example has three coauthors: Ralph Alpher, Hans Bethe, and George Gamow. If the word "and" appears as part of a name, it must be enclosed in braces, as in the example of "Barnes and Noble, Inc." given above. If you have too many names to list in a field, you can end the list with "and others"; the standard styles appropriately append an "et al."

`BIBTEX`'s rules are actually a bit more complicated than indicated here, but this description will suffice for most names.

5.3.4.2. Titles

The bibliography style determines whether or not a title is capitalized; the titles of books usually are, the title of articles usually are not. You type a title the way it should appear if it is capitalized (you should capitalize everything but articles and unstressed conjunctions and prepositions, and even these should be capitalized as the first word or the first after a colon):

```
TITLE = "The Agony and the Ecstasy"
```

`BIBTEX` will change uppercase letters to lowercase if appropriate. Uppercase letters that should not be changed are enclosed in braces. The following two titles are equivalent; the "A" of "Africa" will not be made lowercase.

```
"The Gnats and Gnus of {Africa}" "The Gnats and Gnus of {A}frica"
```

5.3.4.3. Abbreviations

Instead of an ordinary text string, the text of a field can be replaced by an abbreviation for it. An abbreviation is a string of characters that starts with a letter and does not contain a space or any of the following ten characters:

```
" # % ' ( ) , = { }
```

The abbreviation is typed in place of the text field, with no braces or quotation marks. If 'jgg1' is an abbreviation for

```
Journal of Gnats and Gnus, Series~1
```

then the following are equivalent:

```
JOURNAL = jgg1 JOURNAL = "Journal of Gnats and Gnus, Series~1"
```

Some abbreviations are predefined by the bibliography style. These always include the usual 3 letter abbreviations for the month: jan, feb, mar etc.

Bibliography styles usually contain abbreviations for the names of commonly referenced journals. Consult the Local Guide for a list of the predefined abbreviations for the bibliography styles available.

You can define your own abbreviations by using `BIBTEX`'s `@STRING` command.

5.3.5. Field Types

Below is a list of all fields recognized by the standard bibliography styles. An entry can also contain other fields, which are ignored by those styles.

BIBTEX ignores the case of letters in the field names.

5.3.5.1. address field

Publisher's address. For major publishing houses, just the city is given. For small publishers, you can help the reader by giving the complete address.

Format:

```
ADDRESS = field_text
```

5.3.5.2. annote field

An annotation, used only for annotated bibliography styles (which are not among the standard ones).

Format:

```
ANNOTE = field_text
```

5.3.5.3. author field

The name(s) of the author(s).

Format:

```
AUTHOR = field_text
```

5.3.5.4. booktitle field

Title of a book, part of which is being cited.

Format:

```
BOOKTITLE = field_text
```

5.3.5.5. chapter field

A chapter number.

Format:

```
CHAPTER = field_text
```

5.3.5.6. edition field

The edition of a book - for example, "second".

Format:

```
EDITION = field_text
```

5.3.5.7. editor field

Name(s) of editor(s). If there is also an "author" field, then the "editor" field gives the editor of the book or collection in which the reference appears.

Format:

```
EDITOR = field_text
```

5.3.5.8. howpublished field

How something strange has been published.

Format:

```
HOWPUBLISHED = field_text
```

5.3.5.9. institution field

The institution that published the work.

Format:

```
INSTITUTION = field_text
```

5.3.5.10. journal field

A journal name. Abbreviations are provided for many journals; see the Local Guide.

Format:

```
JOURNAL = field_text
```

5.3.5.11. key field

Used for alphabetizing and creating a label when the "author" and "editor" fields are missing. This field should not be confused with the citation key that appears in the `\cite` command and at the beginning of the entry.

Format:

```
KEY = field_text
```

5.3.5.12. month field

The month in which the work was published or, for an unpublished work, in which it was written.

Format:

```
MONTH = field_text
```

5.3.5.13. note field

Any additional information that can help the reader.

Format:

```
NOTE = field_text
```

5.3.5.14. number field

The number of a journal, magazine, or technical report. An issue of a journal or magazine is usually identified by its volume and number; the organization that issues a technical report usually gives it a number.

Format:

```
NUMBER = field_text
```

5.3.5.15. organization field

The organization sponsoring a conference.

Format:

```
ORGANIZATION = field_text
```

5.3.5.16. pages field

A page number or range of numbers such as "42-111"; you may also have several of these, separating them with commas: "7,41,73-97". The standard styles convert a single dash to a double. >

Format:

```
PAGES = field_text
```

5.3.5.17. publisher field

The publisher's name.

Format:

```
PUBLISHER = field_text
```

5.3.5.18. school field

The name of the school where a thesis was written.

Format:

```
SCHOOL = field_text
```

5.3.5.19. series field

The name of a series or set of books. When citing an entire book, the the "title" field gives its title and an optional "series" field gives the name of a series in which the book is published.

Format:

```
SERIES = field_text
```

5.3.5.20. title field

The work's title.

Format:

```
TITLE = field_text
```

5.3.5.21. type field

The type of a technical report - for example, "Research Note".

Format:

```
TYPE = field_text
```


5.3.5.22. `volume` field

The volume of a journal or multivolume book work.

Format:

```
VOLUME = field_text
```

5.3.5.23. `year` field

The year of publication or, for an unpublished work, the year it was written. This field's text should contain only numerals.

Format:

```
YEAR = field_text
```

5.4. `bst` files

Bibliography style files define the style of a bibliography source list.

The standard bibliography style files are `PLAIN`, `UNSRT`, `ALPHA` and `ABBRV`.

If you want to make a bibliography style of your own, look at `SAMPLE.BST`.

5.4.1. `ABBRV.BST`

This style is the same as the style defined in `PLAIN.BST`, except that entries are more compact because first names, month names and journal names are abbreviated.

5.4.2. `ALPHA.BST`

This style is the same as the style defined in `PLAIN.BST` except that entry labels like "Knu66", formed from the author's name and the year of publication, are used.

5.4.3. `PLAIN.BST`

This style is formatted more or less as suggested by Mary-Claire van Leunen in "A Handbook for Scholars" (Alfred A. Knopf, New York, 1979). Entries are sorted alphabetically and are labelled with numbers.

5.4.4. `SAMPLE.BST`

This is a sample bibliography style file meant to help you construct a new style. It creates a bibliography in which entries appear as follows:

```
[Jones79] Jones, R. L. and Richards, P. Q. The Birds and the Bees. {\it Journal of Irreproducible Results 4}, 2 (Jan. 1979), 27-33.
```

```
[Jones82a] Jones, P. G. The Bees and the Trees ... (1982).
```

```
[Jones82b] Jones, R. L. The Trees and the Peas ... (1982).
```

```
[Krist74] Kristofferson, R. P. Peopl and Places ... (1974)
```

It should illustrate how you write a style file. The functions are described in an informal Pascal-like style in comments. Because of the way while loops and if-then-else statements must use functions, the following convention is used. If a while loop is labeled 'foo' in the informal description, then its test and body are the functions named 'foo.test' and 'foo.body'. If an if statement is labeled 'foo', then its two clauses are the functions named 'foo.then' and 'foo.else'. (Null clauses just use the 'skip\$' function.) Note that because functions have to be defined in terms of already-defined functions, the actual function definitions are given in a 'bottom-up' order.

5.4.5. UNSRT.BST

This style is that same as `PLAIN.BST` except that entries appear in the order of their first citation.

6. *MakeIndex*

NOTE

The contents of this section was taken from the HTML helppages for *MakeIndex* of Norman Walsh (Version 1.0, 12 Apr 94).

Some obvious errors in this documentation have been corrected according to `makeindex(1L)`.

Additional information was taken from `makeindex(1L)` and the program source files as found in `teTeX 2.0.2`.

MakeIndex is a general purpose index processor. It takes one or more raw index files (normally generated by a formatter), sorts the entries, and produces the actual index file. It is not dependent on any particular format of raw index file, although the `.idx` file generated by `LATEX` is the default. Up to three levels (0, 1, and 2) of subitem nesting within the same entry is supported. The input format may be redefined in a style file so that raw index or glossary output from other formatters may be processed. The style file also defines the style of output index file. Unless specified otherwise, the file name base of the first input file (`idx0`) is used to determine other related input/output files. The default input file type is `.idx`.

6.1. Options

MakeIndex is a Unix program, and therefore has a Unix-style command line. Instead of qualifiers delimited with a slash (/), *MakeIndex* options are delimited with a hyphen (-).

6.1.1. `-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.

6.1.2. `-g`

Employ German word ordering in the index, in accord with rules set forth in DIN 5007. The quote character must be redefined in a style file (for example, redefine quote as `'+'`). If the quote character is not redefined, *MakeIndex* will produce an error message and abort.

6.1.3. `-i`

Use `stdin` as the input file. When this option is specified and the `-o` is not, output is written to `stdout`.

6.1.4. `-l`

Use letter ordering. Default is word ordering (explained in the Ordering section).

6.1.5. `-q`

Quiet mode, send no messages to `stderr`. By default progress and error messages are sent to `stderr` as well as the transcript file. The `-q` option disables the `stderr` messages.

6.1.6. -r

Disable implicit page range formation. By default three or more successive pages will be automatically abbreviated as a range (e.g. 1-5). The `-r` option disables it, making the explicit range operators the only way to create page ranges (see the Special Effects section below).

6.1.7. -s sty

Take `sty` as the style file. There is no default for the style file name. The environment variable `INDEXSTYLE` defines the path where the style file should be found.

6.1.8. -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.

6.1.9. -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.

6.1.10. -p no

Set the starting page number of the output index file to be `no`. This 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 source log file. The source 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 `[...]`. If a page number is missing or the log file is not found, no attempt will be made to set the starting page number. The meaning of each of these cases follows:

any The starting page is the last source page number plus 1.

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.

6.1.11. -L

Sort based on locale settings. String comparisons for sorting are done using `strcoll(3)`, which compares strings according to the current locale category `LC_COLLATE`.

Not available on all systems (depends on compile time settings).

6.1.12. -T

Special support for Thai documents.

Not available on all systems (depends on compile time settings).

6.2. Style File

The style file format is very simple. It is a list of `<specifier, attribute>` pairs. There are two types of specifiers (input and output). The pairs don't have to obey any particular order in the file. A line lead by '%' is a comment. The following is a list of all the specifiers and their respective arguments where `<string>` is an arbitrary string delimited by double quotes ("`...`"), `<char>` is a single letter embraced by single quotes ('`...`'), and `<number>` is a nonnegative integer. The maximum length of a `<string>` is 144. Notice that a backslash must be escaped (by an extra backslash) in the string quotation. Anything not specified in the style file will be assigned a default value, which is shown on a separate line. This file can reside anywhere in the path defined by the environment variable `INDEXSTYLE`.

6.2.1. Input Style Specifiers

6.2.1.1. `actual <char>`

Default: @

The symbol which indicates that the next entry is to appear in the actual index file.

6.2.1.2. `arg_close <char>`

Default: }

This is the closing delimiter for the index entry argument.

6.2.1.3. `arg_open <char>`

Default: {

This is the opening delimiter for the index entry argument.

6.2.1.4. `encap <char>`

Default: |

The symbol which indicates that the rest of the argument list is to be used as the encapsulating command for the page number.

6.2.1.5. `escape <char>`

Default: \

The symbol which escapes the next letter, unless its preceding letter is escape. In other words, quote is used to escape the letter which immediately follows it. But if it is preceded by escape, it does not escape anything.

Notice that the two symbols must be distinct.

6.2.1.6. `keyword <string>`

Default: "\\indexentry"

This is the command which tells *MakeIndex* that its argument is an index entry.

6.2.1.7. `level <char>`

Default: !

The delimiter which denotes a new level of subitem.

6.2.1.8. quote <char>

Default: "

`quote` is used to escape the letter which immediately follows it, but if it is preceded by escape, it is treated as a ordinary character. These two symbols must be distinct.

6.2.1.9. range_close <char>

Default:)

The closing delimiter indicating the end of an explicit page range.

6.2.1.10. range_open <char>

Default: (

The opening delimiter indicating the beginning of an explicit page range.

6.2.2. Output Style Specifiers**6.2.2.1. page_compositor <string>**

Default: "-"

This specifier is used to separate a range of page numbers.

Officially undocumented!

6.2.2.2. preamble <string>

Default: "\\begin{theindex}\n"

The preamble of actual index file.

6.2.2.3. postamble <string>

Default: "\n\n\\end{theindex}\n"

The postamble of actual index file.

6.2.2.4. setpage_prefix <string>

Default: "\n \\setcounter{page}{"

The prefix of the command which sets the starting page number.

6.2.2.5. setpage_suffix <string>

Default: "}\n"

The suffix of the command which sets the starting page number.

6.2.2.6. group_skip <string>

Default: "\n\n \\indexspace\n"

The vertical space to be inserted before a new group begins.

6.2.2.7. heading_prefix <string>

Default: ""

The header prefix to be inserted before a new letter begins.

6.2.2.8. heading_suffix <string>

Default: ""

The header suffix to be inserted before a new letter begins.

6.2.2.9. headings_flag <string>

Default: 0

The flag indicating the condition of inserting new letter header. Default is 0, which means no header. Positive means insert an uppercase letter between prefix and suffix. Negative means insert a lowercase letter.

6.2.2.10. symhead_positive <string>

Default: "Symbols"

Heading for symbols to be inserted if `headings_flag` is positive.

6.2.2.11. symhead_negative <string>

Default: "symbols"

Heading for symbols to be inserted if `headings_flag` is negative.

6.2.2.12. numhead_positive <string>

Default: "Numbers"

Heading for numbers to be inserted if `headings_flag` is positive.

6.2.2.13. numhead_negative <string>

Default: "numbers"

Heading for numbers to be inserted if `headings_flag` is negative.

6.2.2.14. item_0 <string>

Default: "\n \\item "

The command to be inserted between two primary (level 0) items.

6.2.2.15. item_1 <string>

Default: "\n \\subitem "

The command to be inserted between two secondary (level 1) items.

6.2.2.16. item_2 <string>

Default: "\n \\subsubitem "

The command to be inserted between two level 2 items.

6.2.2.17. item_01 <string>

Default: "\n \\subitem "

The command to be inserted between a level 0 item and a level 1 item.

6.2.2.18. item_x1 <string>

Default: "\n \\subitem "

The command to be inserted between a level 0 item and a level 1 item. The difference between this and previous is that in this case the level 0 item doesn't have any page numbers.

6.2.2.19. item_12 <string>

Default: "\n \\subsubitem "

The command to be inserted between a level 1 item and a level 2 item.

6.2.2.20. item_x2 <string>

Default: "\n \\subsubitem "

The command to be inserted between a level 1 item and a level 2 item. The difference between this and previous is that in this case the level 1 item doesn't have any page numbers.

6.2.2.21. delim_0 <string>

Default: ", "

The delimiter to be inserted between a level 0 key and its first page number. Default is a comma followed by a blank.

6.2.2.22. delim_1 <string>

Default: ", "

The delimiter to be inserted between a level 1 key and its first page number. Default is a comma followed by a blank.

6.2.2.23. delim_2 <string>

Default: ", "

The delimiter to be inserted between a level 2 key and its first page number. Default is a comma followed by a blank.

6.2.2.24. delim_n <string>

Default: ", "

The delimiter to be inserted between two page numbers for the same key in any level. Default is a comma followed by a blank.

6.2.2.25. delim_r <string>

Default: "--"

The delimiter to be inserted between the starting and ending page numbers of a range.

6.2.2.26. `delim_t` <string>

Default: ""

The delimiter to be inserted at the end of a page list. This delimiter has no effect on entries which have no associated page list.

6.2.2.27. `encap_prefix` <string>

Default: "\\\"

The prefix for the command which encapsulates the page number.

6.2.2.28. `encap_infix` <string>

Default: "{"

The prefix for the command which encapsulates the page number.

6.2.2.29. `encap_suffix` <string>

Default: "}"

The suffix for the command which encapsulates the page number.

6.2.2.30. `line_max` <number>

Default: 72

The maximum length of a line in the output beyond which a line wraps around.

6.2.2.31. `indent_space` <string>

Default: "\\t\\t"

The space to be inserted in front of a wrapped line. Default is two tabs.

6.2.2.32. `indent_length` <number>

Default: 16

The length of `indent_space`. In the default case this is 16 (for 2 tabs).

6.2.2.33. `suffix_2p` <string>

Default: ""

Delimiter to replace the range delimiter and the second page number of a two page list. When present, it overrides `delim_r`.

6.2.2.34. `suffix_3p` <string>

Default: ""

Delimiter to replace the range delimiter and the second page number of a three page list. When present, it overrides `delim_r` and `suffix_mp`.

6.2.2.35. suffix_mp <string>

Default: ""

Delimiter to replace the range delimiter and the second page number of a multiple page list (three or more pages). When present, it overrides `delim_r`.

6.3. Example

The following example shows a style file called `book.isty` which defines a stand-alone index for a book. By stand-alone, we mean it can be formatted independent of the main source.

```
preamble
"\documentstyle[12pt]{book}
\begin{document}
\begin{theindex}
{\small\n}

postamble
"\n\n}
\end{theindex}
\end{document}\n"
```

Suppose a particular book style requires the index (as well as any chapters) to start from an odd page number. Given `foo.idx` as the raw index file, the following command line produces an index in file `foo-.ind`.

```
makeindex -s book.isty -o foo-.ind -p odd foo
```

The reason to use a non-default output file name is to avoid clobbering the source output (presumably `foo.dvi`) because if the index is in file `foo.ind`, its output will also be in `foo.dvi` as a result of separate formatting using `.` In the example the index is in `foo-.ind`, its output will be in `foo-.dvi` and thus introduces no confusion.

6.4. Ordering

By default `makeindex` assumes word ordering. The `-l` option turns it into letter ordering. The only difference is whether a blank is treated as an effective letter or not. In word ordering, a blank precedes any letter in the alphabet, whereas in letter ordering, it doesn't count at all. This is best illustrated by the following example:

<i>word order</i>		<i>letter order</i>
sea lion		seal
seal		seal lion

Numbers are sorted in numeric order. For instance,

```
9 (nine), 123
123 10 (ten), see Derek, Bo
```

Letters are first sorted with uppercase and lowercase considered identical; then, within identical words the uppercase letter precedes its lowercase counterpart.

Patterns lead by a special symbol precede numbers, which precede patterns lead by a letter. The symbol here refers to anything not in the union of digits and English alphabet. This includes those which follow 'z' in the ASCII chart. As a special case, anything started with a digit but mixed with non-digits is considered a symbol-leading pattern instead of a number.

6.5. Special Effects

In the normal case entries such as

```
\indexentry{alpha}{1}
\indexentry{alpha!beta}{3}
\indexentry{alpha!beta!gamma}{10}
```

in the raw index file will be converted to

```
\item alpha, 1
  \subitem beta, 3
    \subsubitem gamma, 10
```

in the output index file by *MakeIndex*. Notice that the level symbol (!) is used to delimit levels of nesting.

It is possible to make an item appear in a designated form by using the actual (@) operator. For instance,

```
\indexentry{alpha@{\it alpha\}}{1}
```

will become

```
\item {\it alpha\} 1
```

after the conversion. The idea is that the pattern preceding @ is used as sort key, whereas the one following it is put in the actual result. However, the same key with and without the actual part are regarded as distinct entries.

It is also possible to encapsulate a page number with a designated command using the encap (|) operator. For example, in the default case,

```
\indexentry{alpha|bold}{1}
```

will be converted to

```
\item alpha \bold{1}
```

where `\bold{n}` will expand to `{\bf n}`. This allows the encap operator to be used to set pages in different fonts, thereby conveying more information about whatever being indexed. For instance, given the same key the page where its definition appears can be in one font while where its primary example is given can be in another, with other ordinary appearances in a third. Notice that in this example, the three output attributes associated with page encapsulation `encap_prefix`, `encap_infix`, and `encap_suffix` correspond respectively to backslash, left brace, and right brace. If this is to be formatted by languages other than , they would be defined differently.

By the same token, the `encap` operator can be used to make cross references in the index. For instance,

```
\indexentry{alpha|see{beta}}{1}
```

will become

```
\item alpha \see{beta}{1}
```

in the output index file after the conversion, where

```
\see{beta}{1}
```

will expand to

```
{\it see\} beta
```

Notice that in a cross reference like this the page number disappears. Therefore, where to insert such a command in the source is immaterial.

A pair of encap concatenated with `range_open` (|()) and with `range_close` (|) creates an explicit page range. That is,

```
\indexentry{alpha|(){1}
\indexentry{alpha|)}{5}
```

will become

```
\item alpha, 1--5
```

Intermediate pages indexed by the same key will be merged into the range implicitly. This is especially useful when an entire section about a particular subject is to be indexed, in which case only the range opening and closing operators need to be inserted at the beginning and end of the section, respectively.

This explicit page range formation can also include an extra command to set the page range in a designated font. Thus

```
\indexentry{alpha|bold}{1}
\indexentry{alpha|)}{5}
```

will become

```
\item alpha, \bold{1--5}
```

A couple of special cases are worth mentioning here. First, entries like

```
\indexentry{alpha|}{1}
\indexentry{alpha|bold}{3}
\indexentry{alpha|)}{5}
```

will be interpreted as

```
\item alpha, \bold{3}, 1--5
```

but with a warning message in the transcript about the encounter of an inconsistent page encapsulator. Secondly, an explicit range beginning in a Roman page number and ending in Arabic is considered an error. In a case like this the range is broken into two subranges, if possible, one in Roman, the other in Arabic. For instance,

```
\indexentry{alpha|}{i}
\indexentry{alpha}{iv}
\indexentry{alpha}{3}
\indexentry{alpha|)}{7}
```

will be turned into

```
\item alpha, 1--iv, 3--7
```

with a warning message in the transcript complaining about the illegal range formation.

Finally, every special symbol mentioned in this section may be escaped by the quote operator ("). Thus

```
\indexentry{alpha"@beta}{1}
```

will actually become

```
\item alpha@beta, 1
```

as a result of executing *MakeIndex*. However, if quote is preceded by escape (\), its following letter is not escaped. That is,

```
\indexentry{f\"ur}{1}
```

means

```
\item f\"ur, 1
```

which represents umlaut accented u to the family of processors.

7. xindy

NOTE

The content of this section was taken from the original documentation of xindy V2.1 (`Doc/manual*.html`).

xindy means *flexible indexing system*. It is an indexing system that can be used to generate book-like indexes for arbitrary document preparation systems. This term includes systems such as $\text{T}_{\text{E}}\text{X}$ and $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, the Nroff-family or SGML-based systems (e.g. HTML) that process some kind of text and generate indexing information. It is not fixed to any specific system, but can be configured for a wide variety of purposes.

7.1. Command List

Here is the complete list of xindy's commands that may be used in the index style. The symbol `name` always refers to a string. We separate the commands into the processing and markup commands. The commands are listed in alphabetical order.

The parenthesis [and] denote optional parts of the syntax and { and } denote the grouping of elements. A vertical bar indicates alternatives. However, the enclosing round braces *are* part of the syntax and must be supplied.

7.1.1. Processing Commands

7.1.1.1. `define-alphabet`

```
(define-alphabet name string-list)
```

Defines `name` to be the alphabet consisting of all elements of the `string-list`. Examples:

```
(define-alphabet "example-alphabet" ("An" "Example" "Alphabet"))
```

defines an alphabet consisting of exactly three symbols. For the successor relationship holds: `succ("An")="Example"` and `succ("Example")="Alphabet"`. The built-in alphabet `digits` is defined as follows:

```
(define-alphabet "digits"  
  ("0" "1" "2" "3" "4" "5" "6" "7" "8" "9"))
```

7.1.1.2. `define-attributes`

```
(define-attributes attribute-list)
```

Defines all attributes the raw index may contain. Parameter `attribute-list` is a list of list of strings. The nesting level must not be more than 2. So `(..(..)..)` is allowed, whereas `(..(..(..)..)..)` is not.

The list has two kinds of elements: strings and list of strings. A single string is treated as if it were a single element list. So the lists `("definition")` and `(("definition"))` are equivalent. All elements forming a list are a so-called *attribute group*. The members of a group are written to the output file before any member of the following groups are written.

Examples of valid attributes lists are:

```
("definition" "usage")
```

defines two attribute groups. The first one contains all references with the attribute `definition` and the second one all with the attribute `usage`.

((`"definition" "important"`) `"usage"`) defines two attribute groups. The first one contains all references with the attributes `definition` or `important` and the second one all with the attribute `usage`. In the attribute group (`"definition" "important"`) the attribute `definition` overrides `important`.

7.1.1.3. `define-crossref-class`

```
(define-crossref-class name [:unverified])
```

Defines `name` to be a class of cross references. We distinguish two types of cross reference classes. *Verified* cross reference classes can be checked for dangling references. If for instance a cross reference points to the non-existent keyword ‘foo’ a warning is issued and the user is advised to correct the invalid cross reference. This is the default. If for some reasons this mechanism must be deactivated the switch `:unverified` can be used to suppress this behaviour.

7.1.1.4. `define-letter-group`

```
(define-letter-group name [:before lname] [:after lname]
  [:prefixes list-of-prefixes])
```

```
(define-letter-groups list-of-letter-groups)
```

This command defines a letter group with name `name`, which must be a string value, grouping all index entries that have a *sort key* beginning with the prefix `name`. The command

```
(define-letter-group "a")
```

is equivalent to the command

```
(define-letter-group "a" :prefixes ("a"))
```

Using the latter form one can associate more than one prefix with a given letter group. Also further prefixes can be added to an already existing letter group by simply defining the same letter group again. This results not in a redefinition but in adding more prefixes to the currently defined prefixes.

Example:

```
(define-letter-group "a")
```

defines a letter group containing all index entries beginning with the string "a".

```
(define-letter-group "c" :after "a")
```

defines a letter group containing all index entries beginning with the string "c". The letter group appears behind the letter group "a"

```
(define-letter-group "b" :after "a" :before "c")
```

inserts letter group "b" between letter group "a" and "c". This allows incremental definition of letter groups by extending already defined ones.

The arguments `:after` and `:before` define a partial order on the letter groups. xindy tries to convert this partial order into a total one. If this is impossible due to circular definitions, an error is reported. If more than one possible total ordering can result, it is left open which one is used, so one should always define a complete total order.

The command `define-letter-groups` (with an ‘s’ at the end) is simply an abbreviation for a sequence of `define-letter-group` definitions where the elements are ordered in the ordering given by the list. Example:

```
(define-letter-groups ("a" "b" "c"))
```

equals the definitions

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

See also commands `markup-letter-group-list` and `markup-letter-group` for further information.

7.1.1.5. define-location-class

```
(define-location-class name layer-list
  [:min-range-length num]
  [:hierdepth depth]
  [:var])
```

Defines **name** to be a location class consisting of the given list of layers. A list of layers consists of names of basetypes and/or strings representing separators. Separators must follow the keyword argument **:sep**. If the keyword **:min-range-length** is specified we define the *minimum range length* to be used when building ranges. The argument **num** must be a positive integer number or the keyword **none** in which case the building of ranges is disallowed. If the switch **:var** is specified the declared class is of type *variable*, i.e. it is a *var-location-class*. Since building of ranges is currently only allowed for standard classes **:var** and **:min-range-length** must not be used together. The keyword argument **:hierdepth** can be used to declare that the location references have to be tagged in a hierarchical form. Its argument **depth** must be an integer number indicating the number of layers the hierarchy does contain. See command **markup-locref-list** for more information. Examples:

```
(define-location-class "page-numbers" ("arabic-numbers")
  :minimum-range-length 3)
```

Defines the location class **page-numbers** consisting of one layer which is the alphabet **arabic-numbers**. Since the minimum range length is set to 3 the location references 2, 3 and 4 don't form a range because the range length is only 2. But the references 6, 7, 8, and 9 are enough to form a range. Some example instances of this class are 0, 1, ... 2313, etc.

```
(define-location-class "sections" :var
  ("arabic-numbers" :sep ".")
  ("arabic-numbers" :sep ".")
  ("arabic-numbers"))
```

defines a variable location class. Valid instances are 1, 1.1, 1.2, 2, 2.4.5, but none of 2-3 (wrong separator), 1.2.3.4 (more than 3 layers), 2.3.iv (roman number instead of arabic one).

7.1.1.6. define-location-class-order

```
(define-location-class-order list)
```

Defines the order in which the location classes are written to the output file. The parameter **list** is a list of names of location classes. Examples:

```
(define-location-class-order
  ("page-numbers" "sections" "xrefs"))
```

tells the system that the page numbers should appear before the section numbers and that the cross references should appear at the end. If this command is omitted, the declaration order of the location classes in the index style is implicitly used as the output order. In the case that a location class does not appear in the list, the output may behave unexpectedly, so one should always enumerate all used location classes when using this command.

7.1.1.7. define-rule-set

```
(define-rule-set name
  [ :inherit-from ("rule-set" "rule-set-2") ]
  :rules (<rule>... )
```

A complete specification of a multi-phase sorting process for a language requires that some rules have to appear in several subsequent sorting phases. Rule sets can be used to define a set of rules that can be instantiated in an arbitrary sorting phase. Basically, they offer means to separate the definition of sorting rules from their instantiation, hence, acting as a wrapper for calls to **sort-rule**. They do not add new functionality that is not already present with **sort-rule**.

A rule can be of the form:

```
<rule> ::= ("pattern" "replacement"
           [[:string|:bregexp|:eregexp] [:again]])
```

The following incomplete example defines a new rule set of name `isolatin1-tolower` that inherits definitions from rule set `latin-tolower`, overriding or adding the sort rules in the list of `:rules`.

```
(define-rule-set "isolatin1-tolower"

  :inherit-from ("latin-tolower")

  :rules (("?" "?" :string :again)
          ("?" "?" :string :again)
          ("?" "?" :string :again)
          ("?" "?" :string :again)
          ("?" "?" :string :again)
          ("?" "?" :string :again)
          ("?" "?" :string :again)
          ("?" "?" :string :again)
          ...
         )
  ...)
```

Rule sets can be instantiated with the command `use-rule-set`. For further descriptions on the sorting model refer to the command `sort-rule`.

7.1.1.8. `define-sort-rule-orientations`

```
(define-sort-rule-orientations (orientations...))
```

Defines the order for the different sorting phases. The currently implemented *orientations* are `forward` and `backward`. This command must precede all `sort-rule` commands in an index style. It defines the orientations and implicitly sets the maximum number of sorting phases performed.

For further descriptions on the sorting model refer to the command `sort-rule`.

7.1.1.9. `merge-rule`

```
(merge-rule pattern replacement [:again]
           [[:bregexp | :eregexp | :string]])
```

Defines a keyword mapping rule that can be used to generate the *merge key* from the *main key* of an index entry. This mapping is necessary to map all keywords that are differently written but belong to the same keyword to the same canonical keyword.

The parameter `pattern` can be a POSIX-compliant regular expression or an ordinary string. The implementation uses the GNU Rx regular expression library which implements the POSIX regular expressions. Regular expressions (REs) can be specified as *basic regular expressions* (BREs) or *extended regular expressions* (EREs). You can use the switch `:bregexp` to force the interpretation of the pattern as a BRE, or `:eregexp` to interpret it as an ERE. If you want xindy to interpret the pattern literally, use the switch `:string`. If none of these switches is selected, xindy uses an auto-detection mechanism to decide, if the pattern is a regular expression or not. If it recognizes the pattern as a RE, it interprets it as an ERE by default.

The parameter `replacement` must be a string possibly containing the special characters `&` (substitutes for the complete match) and `\1` , . . . , `\9` (substituting for the *n*-th submatch. Examples:

```
(merge-rule "A" "a")
```

replaces each occurrence of the uppercase letter ‘A’ with its lowercase counterpart.

```
(merge-rule "\~"("[AEUOaeou])" "\1")
```

transforms the \TeX umlaut-letters into their stripped counterparts, such that ‘\A ’ is treated as an ‘A’ afterwards.

The following sequences have a special meaning:

- ‘ `~n` ’ End of line symbol (*linefeed*).
- ‘ `~b` ’ The ISO-Latin character with the lowest ordinal number.
- ‘ `~e` ’ The ISO-Latin character with the highest ordinal number.
- ‘ `~` ’ The tilde character.
- ‘ `~"` ’ The double quote character.

Tilde characters and double quotes have to be quoted themselves with a tilde character. The special characters ‘ `~b` ’ and ‘ `~e` ’ allow the definition of arbitrary sorting orders by rules. In connection with an additional character every position in the alphabet can be described. E.g. ‘ `m~e` ’ is lexicographically placed between ‘ `m` ’ and ‘ `n` ’.

Due to efficiency, rules that just exchange characters or substitute constant character sequences are not treated as regular expressions. Therefore, instead of using the rule

```
(merge-rule "[A-Z]" "&")
```

it is more efficient (though less comfortable) to use

```
(merge-rule "A" "Ax")
(merge-rule "B" "Bx")
...
(merge-rule "Z" "Zx")
```

Usually rules are applied in order of their definition. Rules with a special prefix precede those that begin with a class of characters, so that the search pattern ‘ `alpha` ’ is checked before ‘ `.*` ’, but ‘ `auto` ’ and ‘ `a.*` ’ are checked in order of their definition.

The first rule from a style file that matches the input is applied and the process restarts behind the substituted text. If no rule could be applied, the actual character is copied from the input and the process continues with the next character.

Sometimes it is necessary to apply rules anew to the result of a transformation. By specifying the keyword argument `:again` in the merge rule the rule is marked as mutable, which means that after using this rule the transformation process shall restart at the same place. E.g. the rule

```
(merge-rule "\$(.*)\$" "\1" :again)
```

deletes *all* surrounding ‘ `$` ’ symbols from the input.

See also command `sort-rule`.

7.1.1.10. `merge-to`

```
(merge-to attr-from attr-to [:drop])
```

A merge rule says that the attribute `attr-from` can be used to build ranges in `attr-to`. Both attributes must name valid attribute names. The switch `:drop` indicates, that the original location reference with attribute `attr-from` has to be dropped (removed), if a successful range was built with location references in attribute `attr-to`. A detailed description is given in the section about processing phases.

7.1.1.11. `require`

```
(require filename)
```

This command allows to load more index style modules. The module is searched in the directories defined in the search path. The file is read in and processing of the current file continues. The argument `filename` must be a string. This allows to decompose the index style into several modules that can be included into the topmost index style file. Example:

```
(require "french/alphabet.xdy")
(require "french/sort-rules.xdy")
```

```
(require "tex/locations.xdy")
(require "tex/markup.xdy")
```

Submodules can load other submodules as well. If a file is required that was already loaded, the `require` command is simply ignored and processing continues without including this file twice. See also command `searchpath`.

7.1.1.12. searchpath

```
(searchpath {path-string | path-list})
```

This command adds the given paths to the list of paths, xindy searches for index style files. The argument `path-string` must be a colon-separated string of directory names. If this path ends with a colon the default search path is added to the end of the path list. Example:

```
(searchpath " ./usr/local/lib/xindy:/usr/local/lib/xindy/english:")
```

adds the specified directories to the search path. Since the last path ends with a colon, the built-in search path is added at the end. Specifying

```
(searchpath ( "."
              "/usr/local/lib/xindy"
              "/usr/local/lib/xindy/english"
              :default))
```

yields exactly the same result as the example above. Here `path-list` must be a list of strings and/or the keyword(s) `:default` and `:last`. The keyword `:default` signifies that the default pathnames are to be inserted at the specified position in the list. The keyword `:last` allows to insert the currently active paths at the indicated position. Since this allows to insert the built-in paths at any position and incrementally adding new paths to the search path, this version of the command is more flexible than the first version.

7.1.1.13. sort-rule

```
(sort-rule pattern replacement [:run level] [:again])
```

Defines a keyword mapping rule that can be used to generate the *sort key* of an index entry from the *merge key*. This key is used to sort the index entries lexicographically after they have been merged using the merge key.

The argument `:run` indicates that this rule is only in effect at the specified *level* (default is level 0). For a detailed discussion on the definition of sort rules for different layers refer to the documentation about the new sorting scheme (`new-sort-rules`) that comes with this distribution.

See command `merge-rule` for more information about keyword rules.

7.1.1.14. use-rule-set

```
(use-rule-set [:run phase]
              [:rule-set ( <rule-set>... )])
```

This command instantiates the gives rule sets to be in effect at sorting phase `phase`. The order of the rule sets given with argument `:rule-set` is significant. Rule set entries of rule set appearing at the beginning of the list override entries in rule sets at the end of the list.

The following example declares that in phase 0 the rule sets `din5007` and `isolatin1-tolower` should be active, whereas in phase 2 the other rule sets have to be applied.

```
(use-rule-set :run 0
              :rule-set ("din5007" "isolatin1-tolower"))

(use-rule-set :run 1
              :rule-set ("resolve-umlauts"
                        "resolve-sharp-s"
                        "isolatin1-tolower"
                        ))
```

For a discussion on rule sets refer to command `define-rule-set`.

7.1.2. Markup Commands

The following commands can be used to define the markup of the index. They don't have any influence on the indexing process. Since the markup scheme is characterized by the concept of *environments*, the syntax and naming scheme of all commands follows a simple structure.

The commands can be separated into *environment* and *list-environment* commands. All commands of the first group support the keyword arguments `:open` and `:close`, whereas the second group additionally supports the keyword argument `:sep`. If one of these keyword arguments is missing, the default markup tag is always the empty tag. The `:open` tag is always printed before the object itself and the `:close` tag is always printed after the object has been printed. If a list is printed the `:sep` tag is printed between two elements of the list but not before the first element, or after the last one. All commands dealing with a list have the suffix '`-list`' as part of their command name.

Since the number of commands and the heavy usage of *default* and *specialized* tags makes the markup somehow complex (but very powerful) we have added a mechanism to trace the markup tags xindy omits during its markup phase with the command `markup-trace`.

Here follows the list of markup commands in alphabetical order with some of the commands grouped together.

7.1.2.1. markup-attribute-group-list

```
(markup-attribute-group-list [:open string] [:close string]
                             [:sep string])
```

```
(markup-attribute-group      [:open string] [:close string]
                             [:group group-num])
```

Location class groups consist of lists of attribute groups. The markup of this list can be defined with the command `markup-attribute-group-list`.

To allow different markup for different attribute groups the command `markup-attribute-group` can be specialized on the group number with the keyword argument `:group` which must be an integer number. E.g., given are the groups ("definition" "theorem") and ("default") with group numbers 0 and 1, then

```
(markup-attribute-group :open "<group0>" :close "</group0>"
                       :group 0)
```

```
(markup-attribute-group :open "<group1>" :close "</group1>"
                       :group 1)
```

can be used to assign different markup for both groups in a SGML-based language.

7.1.2.2. markup-crossref-list

```
(markup-crossref-list      [:open string] [:close string]
                           [:sep string]
                           [:class crossref-class])
```

```
(markup-crossref-layer-list [:open string] [:close string]
                            [:sep string]
                            [:class crossref-class])
```

```
(markup-crossref-layer     [:open string] [:close string]
                           [:class crossref-class])
```

A crossref class group contains cross references of the same class. The separator between the classes is defined with the `(markup-locclass-list :sep)`-parameter. A list of cross references can be tagged with the command `markup-crossref-list` that specializes on the `:class` argument.

Each cross reference is determined by a list of layers indicating the target of the cross reference. To define a suitable markup for such a list the command `markup-crossref-layer-list` can be used.

Each layer of a cross reference can be assigned two tags that specialize on the class of the reference, like all other commands.

A suitable markup for a cross reference class `see` within $\text{\LaTeX} 2_{\epsilon}$ could look like that:

```
(markup-crossref-list :class "see" :open "\emph{see} "
                    :sep "; ")
(markup-crossref-layer-list :class "see" :sep ",")
(markup-crossref-layer :class "see"
                      :open "\textbf{" :close "}")
```

An example output could look like ... `see house; garden,winter; greenhouse`

7.1.2.3. markup-index

```
(markup-index [:open string] [:close string]
             [ :flat | :tree | :hierdepth depth ])
```

Defines the markup tags that enclose the whole index via the `:open` and `:close` parameters. Examples:

```
(markup-index :open "Here comes the index~n"
             :close "That's all folks!~n")
```

defines that the `:open` string is printed before the rest of the index and the `:close` string appears after the index is printed.

Additionally one can specify the form of the generated index. It is possible to produce flat indexes by specifying the switch `:flat`, to generate a tree with the `:tree` switch or any kind of mixture between both by specifying the depth up to which trees shall be built with the parameter `:hierdepth`. Its argument depth is the number of layers that can be formed into a tree. Therefore `:flat` is an abbreviation of `:hierdepth 0` and `:tree` is an abbreviation of `:hierdepth max-depth`, with `max-depth` being the maximum number of layers a keyword has. An example: the keywords

```
("tree" "binary" "AVL")
("tree" "binary" "natural")
```

can be transformed in the following ways:

A flat index (`:flat` or `:hierdepth 0`)

```
tree binary AVL
tree binary natural
```

with `:hierdepth 1`

```
tree
  binary AVL
  binary natural
```

and a tree (`:tree` or `:hierdepth > 1`)

```
tree
  binary
    AVL
    natural
```

Most often one will create tree-like indexes or ones that are flat.

7.1.2.4. markup-indexentry-list

```
(markup-indexentry-list [:open string] [:close string]
                       [:sep string] [:depth integer])
```

```
(markup-indexentry      [:open string] [:close string]
                       [:depth integer])
```

Letter groups consists of a list of index entries. The command `markup-indexentry-list` defines the markup of these lists. The markup can be specialized on the depth if the index is hierarchically organized. The command

```
(markup-indexentry-list :open "\begin{IdxentList}"
                        :close "\end{IdxentList}"
                        :sep "~n")
```

defines that the index entries of all layers are wrapped into the given markup tags. If additionally

```
(markup-indexentry-list :open "\begin{IdxentListII}"
                        :close "\end{IdxentListII}"
                        :sep "~n"
                        :depth 2)
```

is defined, all index entry lists of all layers (except layer 2) are tagged according to the first specification, and the index entry list within depth 2 are tagged according to the second rule.

The command `markup-indexentry` defines the markup of an index entry at a given depth. Since index entries may also contain subentries and the markup for subentries may be different in different layers, the optional keyword argument `:depth` can be used to assign different markup for different layers. If depth is omitted the default markup for all possible depths is defined. The top-most index entries have depth 0.

```
(markup-indexentry :open "\begin{Indexentry}"
                  :close "\end{Indexentry}")
```

defines that the index entries of all layers are wrapped into the given markup tags. If additionally

```
(markup-indexentry :open "\begin{IndexentryII}"
                  :close "\end{IndexentryII}"
                  :depth 2)
```

is defined, all index entries of all layers (except layer 2) are tagged according to the first specification, and the index entries with depth 2 are tagged according to the second rule.

7.1.2.5. markup-keyword-list

```
(markup-keyword-list [:open string] [:close string]
                    [:sep string] [:depth integer])
```

```
(markup-keyword      [:open string] [:close string]
                    [:depth integer])
```

The print key of an index entry consists of a list of strings. The markup of this list can be defined with the command `markup-keyword-list`. The keyword argument `:depth` may be specified to define the markup of the list at a particular depth.

The keyword of an index entry consists of a list of strings. Each of these components is tagged with the strings defined with the command `markup-keyword`. Since we maybe need different markup for different layers, the optional keyword argument can be used to specialize this markup for some depth.

7.1.2.6. markup-letter-group-list

```
(markup-letter-group-list [:open string] [:close string]
                         [:sep string])
```

```
(markup-letter-group  [:open string] [:close string] [:group group-name]
                     [:open-head string] [:close-head string]
                     [:upcase | :downcase | :capitalize])
```

The first command defines the markup of the letter group with name `group-name`. Since the markup of letter groups often contains the name of the letter group as a part of it, the other keyword arguments allow an additional markup for this group name. If one of the parameters `:open-head` and `:close-head` is specified additional markup is added as can be described as follows:

```

<OPEN>
  IF (:open-head OR :close-head)
    <OPEN-HEAD>
      transformer-of(<GROUP-NAME>)
    <CLOSE-HEAD>
  FI
  <INDEXENTRIES...>
<CLOSE>

```

Here, `transformer-of` is a function that possibly transforms the string representing the group name into another string. The transformers we currently support can be specified with the switches `:uppercase`, `:downcase` and `:capitalize` which result in the corresponding string conversions. If none of them is specified no transformation is done at all.

The command `markup-letter-group` defines the markup of the list of letter groups.

7.1.2.7. `markup-locclass-list`

```

(markup-locclass-list [:open string] [:close string]
  [:sep string])

```

Each index entry contains a list of location class groups. This markup command can be used to define the markup of this list.

7.1.2.8. `markup-locref`

```

(markup-locref [:open string] [:close string]
  [:class locref-class]
  [:attr attribute]
  [:depth integer])

```

The markup tags of a location reference can be specialized on the three arguments `:class`, `:attr` and additionally, if sub-references are used, `:depth`. Most often one will only use a tag depending on the attribute. For example, all location references with the attribute `definition` should appear in a font series like bold, emphasizing the importance of this location reference; those with the attribute `default` in font shape italic. The markup in this case would not specialize on the depth or any particular class. A valid definition, suitable for a usage within HTML, could look like this.

```

(markup-locref :open "<B>" :close "</B>" :attr "definition")
(markup-locref :open "<I>" :close "</I>" :attr "default")

```

7.1.2.9. `markup-locref-class`

```

(markup-locref-class [:open string] [:close string]
  [:class locref-class])

```

All location references of a particular location reference class can be wrapped into the tags defined by this command. It specializes on the keyword argument `:class`.

7.1.2.10. `markup-locref-layer`

```

(markup-locref-layer      [:open string] [:close string]
  [:depth integer] [:layer integer]
  [:class locref-class])

```

```

(markup-locref-layer-list [:open string] [:close string]
  [:sep string]
  [:depth integer]
  [:class locref-class])

```

A location reference contains a list of location reference layers. The second markup command can be used to markup this list. It specializes on the class of the location references and the depth (if sub-references are used).

The first command allows to tag the elements of a layer list differently. The first element of this list can be specialised with `:layer 0`, the next element with `:layer 1`, etc. See the next example for an example.

7.1.2.11. markup-locref-list

```
(markup-locref-list [:open string] [:close string] [:sep string]
                  [:depth integer] [:class locref-class])
```

An attribute group contains a list of location references and/or ranges. Additionally a layered location reference itself may contain sub-references that are stored as a list of location references. We specialize the markup for these lists on the location class they belong to with the keyword argument `:class`, and on `:depth` that specializes on the different subentry levels when using location references with sub-references.

Given is a list of location references that have the class description

```
(define-location-class "Appendix"
  ("ALPHA" :sep "-" "arabic-numbers")
  :hierdepth 2)
```

This location class has instances like A-1, B-5, etc. The keyword argument `:hierdepth 2` informs xindy to markup these location references in a hierarchical form. With the commands

```
(markup-locref-list          :sep ";" "
  :depth 0 :class "Appendix")
(markup-locref-list :open " " :sep ", "
  :depth 1 :class "Appendix")
(markup-locref-layer :open "{\bf " :close "}" :layer 0
  :depth 0 :class "Appendix")
```

we obtain a markup sequence for some example data that could look like

```
\bf A} 1,2,5; {\bf B} 5,6,9; {\bf D} 1,5,8; ...
```

7.1.2.12. markup-range

```
(markup-range [:open string] [:close string] [:sep string]
              [:class locref-class]
              [:length num] [:ignore-end])
```

A range consists of two location references. Markup can be specified with the `:open` and `:close` arguments and one separator given by the argument `:sep`.

Since both location references are tagged with markup defined by the command `markup-locref` a specialization on attributes or depth is not necessary. Specialization is allowed on the class they belong to, because the separator between two location references may be different for each location class. Argument `:length` can be used to define different markup for different lengths. In conjunction with `:length` it may be useful not to print the second location reference at all. For example, one wishes to markup ranges of length 1 in the form `Xf`. instead of `X–Y`. This can be accomplished with the switch `:ignore-end`.

The markup tags for a range (X,Y) can be described as follows:

```
<OPEN>
  Markup of location reference X
<SEP>
  IF (not :ignore-end)
    Markup of location reference Y
  FI
<CLOSE>
```

The following tags can be used to define a range of page numbers (given in a location class `page-numbers`) without considering the open and close parameters:

```
(markup-range :sep "-" :class "page-numbers")
```

Location ranges then appear separated by a hyphen in a form like this:

```
..., 5-8, 19-23, ...
```

7.1.2.13. (markup-trace [:on] [:open string] [:close string])

This command can be used to activate the tracing of all markup commands xindy executes. The switch `:on` activates the trace. If `:on` is omitted, the command line flag `-t` can be used as well. All tags which are emitted but not yet defined explicitly by the user are tagged with a symbolic notation indicating the commands that must be used to define this tag. The defaults for the keyword argument `:open` is `'<'` and for `:close` is `'>'`. The beginning of an example output could look like:

```
<INDEX:OPEN>
  <LETTER-GROUP-LIST:OPEN>
    <LETTER-GROUP:OPEN ["a"]>
      <INDEXENTRY-LIST:OPEN [0]>
        <INDEXENTRY:OPEN [0]>
          <KEYWORD-LIST:OPEN [0]>
            <KEYWORD:OPEN [0]>
    ...
```

We use a simple indentation scheme to make the structure of the tags visible. The symbolic tag `<LETTER-GROUP:OPEN ["a"]>` for example indicates that the tag that can be specified with the command

```
(markup-letter-group :open "XXX" :group "a" ... )
```

is emitted at this point in the markup process. By incrementally adding markup commands to the index, more and more tags can be defined until the whole markup is defined. This general mechanism should allow everyone understand the markup process. The best is to start with a small index, define the complete markup and afterwards process the whole index. Additionally one can enclose the symbolic tags into an environment that is neutral to the document preparation system, such as a comment. For TeX this could be

```
(markup-trace :open "%%" :close "~n")
```

or a definition in the TeX document like

```
\def\ignore#1{}
```

combined with the command

```
(markup-trace :open "\ignore{" :close "}")
```

7.1.3. Raw Index Interface

This section can be skipped if the reader is not interested in adapting xindy to a new document preparation system.

The raw index is the file that represents the index that is to be processed. Since many different document preparation systems may use different forms of index representations, their output must be transformed in a form readable by xindy. We also could have written a configurable parser performing this task, but usually a tool written with some text processing tools such as **perl**, **sed** or **awk** can achieve the same task as well. Therefore, adapting xindy to a completely different system can mostly be done by writing an appropriate raw index filter.

The format of the raw index interface of xindy is defined as follows:

```
(indexentry { :key string-list [:print string-list]
  | :tkey list-of-layers }
  [:attr string]
  { :locref string [:open-range | :close-range]
  | :xref string-list } )
```

The pseudo variable *string* is a sequence of characters surrounded by double quotes, e.g.


```
"Hi, it's me" "one" "a string with two \"double quotes\""
```

are three examples of valid strings. If you need to include a double quote as a literal character, you must quote it itself with a backslash as shown in the third example. A *string list* is simply a list of strings separated by whitespaces and surrounded by round braces. An example of a string list is

```
("This" "is" "a" "list" "of" "strings")
```

So far about the syntax. The semantics of the different elements are described here.

:key The argument *string list* defines the keyword of the index entry. It must be a list of strings, since the keyword may consist of different layers such as ("heap" "fibonacci").

:print The optional *print key* defines the way the keyword has to be printed in the markup phase.

:tkey Another possibility to define the keys of an index entry is with the **:tkey** keyword argument. It can be used instead of the **:key** and **:print** arguments. Instead of specifying separately the key and the corresponding print key, we define the keyword by its layers. Each layer consist of a list of one or two strings. The first string will be interpreted as the main key, whereas the second one will become the print key. If the print key is omitted, the main key is taken instead. So the definition

```
:tkey (("This") ("is") ("a") ("bang" "BANG !!!"))
```

is equivalent to

```
:key ("This" "is" "a" "bang")
```

```
:print ("This" "is" "a" "BANG !!!")
```

:locref The reference an index entry describes can be a *location reference* or a *cross reference*. The switch **:locref** describes a location reference. Its optional arguments are **:open-range** and **:close-range**. The *string* that must be supplied must somehow encode the location reference. It might look like the string "25" representing the page number 25, or "Appendix-I" representing the first appendix numbered in uppercase roman numerals.

:open-range, **:close-range** These are switches that do not take any arguments. They describe the beginning and ending of a *range*, starting or ending from the location reference that is given by the argument **:locref**. If they are supplied, the location reference may have influence on the way ranges are build.

:xref These arguments choose the second alternative. The argument *string list* of parameter **:xref** describes where the index entry should point to.

:attr This parameter may be used to tag a location reference with a certain attribute or it names the class of a cross reference. It may also used to associate different markup for different attributes in the markup phase. If this parameter is omitted or is the empty string, the indexentry is declared to have the attribute **default**.

Some examples:

```
(indexentry :key ("airplane") :locref "25" :attr "default")
```

defines an index entry with the key *airplane* indexed on page 25. This index entry has the attribute **default**.

```
(indexentry :key ("house") :xref("building") :attr "see")
```

defines a cross reference with the key *house* pointing to the term *building*. This cross reference belongs to the cross reference class *see*.

```
(indexentry :key ("house") :xref("building") :open-range)
```

is an invalid specification, since **:open-range** mustn't be used together with cross references.

7.2. Invoking xindy

7.2.1. Command Line Options

The following command line options are accepted:

```
xindy [-h] [-t] [-v] [-l logfile] [-o outfile]
      [-L n] [-f filterprog]
      indexstyle raw-index
```

The argument `indexstyle` names a file, containing the index style description. The argument `raw-index` names a file, containing the raw index. Both arguments are mandatory.

- h** Gives a short summary of all command line options.
- l** Writes helpful information into the specified `logfile`. For example, the keyword mappings are written into this file, so one can check if the intended mappings were actually performed this way.
- o** Explicitly defines the name of the `output` file. If not given, the name of the `raw-index` is used with its extension changed to `.ind` (or added, if it had no extension at all).
- t** Enters tracing mode of the symbolic markup tags. The format of the emitted tags can be defined with the command `markup-trace`.
- L** Set the xindy logging-level to `n`.
- f** Run `filterprog` on `raw-index` before reading. The program must act as a filter reading from `stdin` and writing to `stdout`. The most obvious use of this option in conjunction with TeX is to run `-f tex2xindy` on the index file prior to reading the entries into xindy.
- v** Shows the version number of xindy.

Errors and warnings are reported to `stdout` and additionally to the logfile if `-l` was specified.

7.2.2. Search Path

The system uses the concept of a search path for finding the index style files and modules. The searchpath can be set with the environment variable `XINDY_SEARCHPATH` which must contain a list of colon-separated directories. If it ends with a colon, the built-in searchpath is added to the entire searchpath. See the command `searchpath` for further details.

A. Appendices

A.1. Bugs / Known Issues / Missing Features

PDF The PDF file is an experimental version generated using DB2 \LaTeX <<http://dblatex.sf.net>>. There is much room for improvements and we will work on this.

Texinfo Currently we don't know any working solution to convert DocBook XML to Texinfo (at least not for a complex document). Anyone who wants to work on this issue is highly welcome. A starting point could be the *docbook2X* project at <<http://docbook2x.sourceforge.net>>.

vim-help Quite a few people are asking for a version in vim-help format. Srinath Avadhanula <<mailto:srinath@eecs.berkeley.edu>> has already done some work on this.

A.2. About this Document

The source format of this document is DocBook XML (V4.2) <<http://www.oasis-open.org/committees/docbook/>>.

The various output formats were generated using:

- xsltproc (libxml2-2.6.10/libxslt-1.1.7) <<http://xmlsoft.org/XSLT/>> as XSLT processor
- DocBook XSL Stylesheets (V1.65.0) <<http://sourceforge.net/projects/docbook>> for HTML
- DB2 \LaTeX XSL Stylesheets (V0.8pre1+20040621 <<http://db2latex.sf.net/>> for \LaTeX /PDF
- scrbook [2004/04/26 v2.9q LaTeX2e KOMA document class] of KOMA-Script <<ftp://dante.ctan.org/tex-archive/macros/latex/contrib/koma-script/>>
- Latin Modern Type 1 Fonts (V0.92) <<ftp://ftp.dante.de/tex-archive/fonts/ps-type1/lm/tex/latex/lm/>>

You'll always find the newest version of this document at <http://www.miwie.org/tex-refs/> <<http://www.miwie.org/tex-refs/>>.

A.3. Release News

- V0.3.6 04-06-21
- Added a note in chapter \LaTeX with the copyright and permission notice of the original file `latex2.texi`
 - Corrected typos and added missing hyperlinks in chapter `xindy`
 - PDF file optimized/linearized and some finetuning
- V0.3.5 04-05-01
- Added content to chapters `MetaPost` and `pdf \LaTeX`
 - Enhanced index for chapter *MakeIndex*
- V0.3.4 04-04-24
- Version control system changed from CVS to SVN
 - Added missing entries to chapter *MakeIndex*
- V0.3.3 04-01-26
- Corrected again section `Output Style Specifiers` and `Input Style Specifiers` of chapter *MakeIndex*
 - Corrected some symbols in the index

- V0.3.2 04-01-16
 - Corrected sections Output Style Specifiers and Input Style Specifiers of chapter *Make-Index*
 - Enhanced index in PDF
- V0.3.1 04-01-06
 - More work on PDF
 - Modularized XSL stylesheet files
 - Corrected some typos
- V0.3.0 03-12-25
 - Changed document root to *book*
 - More work on PDF version
 - Using new XSL stylesheets V1.64.1
 - Minor fixes
- V0.2.6 03-10-27
 - Rearranged appendix
 - Work on PDF version
- V0.2.5 03-10-17
 - Added contents of GNU FDL in appendix
 - Using new XSL stylesheets V1.62.4
 - Switched to `xsltproc` <<http://xmlsoft.org/XSLT/>> as XSLT-processor
 - Reedited some deeply nested sections
 - Changed colors in CSS
- V0.2.4 03-09-06
 - Added subsection `dcolumn`
 - Subsection `fontenc` translated into english
 - Corrected typo in `\sbox`
- V0.2.3 03-04-12
 - Provide PDF output using ConTeXt <<http://www.pragma-ade.com/>> and Doc-Book In ConTeXt <<http://www.hobby.nl/~scaprea/context/index.html>> (still experimental)
 - Reedited (beautified) sections LaTeX / Commands / Counters | Cross References | Definitions | Layout | Environments | Footnotes
- V0.2.2 03-01-26
 - Added bzip2 compressed version of source and outfiles tarball
 - Using new XSL stylesheets V1.60.1
 - Eliminated more spurious `Â` characters in HTML output
- V0.2.1 03-01-18
 - Using new XSL stylesheets V1.59.2
 - Eliminated spurious `Â` characters
- V0.2.0 03-01-11
 - Using new XSL stylesheets V1.58.1
 - Minor changes to CSS file
 - License changed to GNU Free Documentation License <<http://www.gnu.org/licenses/fdl.html>>
 - Added subsection Commands and Parameters to section PSTricks
 - Reworked `inputenc` section
- V0.1.3 02-10-17
 - Started rework of KOMA-Script section
 - Added subsection Additional PSTricks Packages
 - Using new XSL stylesheets V1.56.1
- V0.1.2 02-10-01
 - Reworked `hyperref` section
- V0.1.1 02-09-19
 - Added template sections for `PiCTEX` and `Texinfo`

- Added subsection Release News in appendix About this Document
 - Using new XSL stylesheets V1.55.0
 - Added missing CSS file to outfiles tarball
 - Corrected wrong FPI
- V0.1.0 02-08-20
- Eliminated trailing '.' in numbered sections
 - Minor markup errors corrected
 - Reworked CSS file
- V0.0.5 02-08-10
- Switched to DocBook XML 4.2
 - Sections 'Bibindex' and 'xindy' completed
- V0.0.4 02-07-25
- Section 'Makeindex' completed
- V0.0.3 02-07-13
- New XSL stylesheets solve bug in creating index (no other changes)
- V0.0.2 02-07-05
- New (template) sections 'fontinst', 'Bibtex', 'Makeindex', and 'xindy'
 - New section 'Special Symbols'
 - Added information for 'german' package
 - Registered as a new project (tex-refs) on freshmeat.net
- V0.0.1 02-06-21
- First official announcement

A.4. GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc.
 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by

proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit

reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions

it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of

Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual

title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

Symbols

\$	29	\encl	26
~	29	\end	13
^	29	\enlargethispage	27
\!	30	\ensuremath{}	30
\' (tabbing)	20	\fbox	33
\(29	\flushbottom	13
\)	29	\fnsymbol	10
\+	20	\fontencoding	45
\,	30	\fontfamily	45
\- (hyphenation)	27	\fontseries	45
\- (tabbing)	20	\fontshape	45
\;	30	\fontsize	45
\=	20	\footnote	24
\Alph	10	\footnotemark	24
\COLON	30	\footnotesize	44
\Huge	44	\footnotetext	24
\LARGE	44	\frac	30
\Large	44	\frame	19
\Roman	10	\framebox	19, 33
\[29	\fussy	28
\]	27	\graphicspath	47
\>	20	\height	25
\<	20	\hfill	33
\]	29	\hline	22
\' (tabbing)	20	\hrulefill	33
\a	20	\hspace	33
\addcontentsline	43	\huge	44
\address	26	\hyphenation	28
\addtocontents	43	\include	42
\addtocounter{counter}{value}	10	\includegraphics	47
\addtolength	25	\includeonly	42
\addvspace	33	\indent	29
\alph	10	\input	42
\appendix	32	\item	17
\arabic	10	\itshape	44
\areaset	51	\kill	20
\author	31	\label	11
\backslash	35	\large	44
\begin	13	\ldots	30
\bfseries	44	\lefteqn	15
\bibitem	23	\line	19
\bibliography	23	\linebreak	28
\bibliographystyle	23	\linethickness	19
\bigskip	33	\listoffigures	43
\bm	60	\listoftables	43
\boldsymbol	45	\location	26
\boldsymbol	60	\makebox	34
\caption	16	\makebox (picture)	19
\cc	26	\makelabels	26
\cdots	30	\maketitle	31
\centering	14	\markboth	31
\chapter	32	\markright	32
\circle	18	\mathbf	44
\cite	23	\mathcal	44
\cleardoublepage	27	\mathit	44
\clearpage	27	\mathnormal	44
\cline	21	\mathrm	44
\closing	26	\mathsf	44
\dashbox	18	\mathtt	44
\date	31	\mathversion	44
\ddots	30	\mbox	34
\depth	25	\mdseries	44
\documentclass	12	\medskip	33
\dotfill	32	\multicolumn	22
\emph	44	\multitup	19
		\name	26

<code>\newcommand</code>	11	<code>\subsubsection</code>	32
<code>\newcounter</code>	10	<code>\symbol</code>	35
<code>\newenvironment</code>	11	<code>\tableofcontents</code>	43
<code>\newfont</code>	12	<code>\telephone</code>	27
<code>\newlength</code>	25	<code>\textbf</code>	44
<code>\newline</code>	28	<code>\textit</code>	44
<code>\newpage</code>	28	<code>\textmd</code>	44
<code>\newsavebox</code>	34	<code>\textnormal</code>	44
<code>\newtheorem</code>	12	<code>\textrm</code>	44
<code>\nocite</code>	23	<code>\textsc</code>	44
<code>\nofiles</code>	43	<code>\textsf</code>	44
<code>\noindent</code>	29	<code>\textsl</code>	44
<code>\nolinebreak</code>	28	<code>\texttt</code>	44
<code>\nopagebreak</code>	28	<code>\textup</code>	44
<code>\normalfont</code>	44	<code>\thanks</code>	31
<code>\normalsize</code>	44	<code>\thispagestyle</code>	32
<code>\onecolumn</code>	13	<code>\tiny</code>	44
<code>\opening</code>	26	<code>\title</code>	31
<code>\oval</code>	19	<code>\today</code>	24
<code>\overbrace</code>	30	<code>\totalheight</code>	25
<code>\overline</code>	30	<code>\ttfamily</code>	44
<code>\pagebreak</code>	28	<code>\twocolumn</code>	13
<code>\pagenumbering</code>	31	<code>\typearea</code>	51
<code>\pageref</code>	11	<code>\typein</code>	43
<code>\pagestyle</code>	31	<code>\typeout</code>	43
<code>\par</code>	29	<code>\unboldmath</code>	60
<code>\paragraph</code>	32	<code>\underbrace</code>	30
<code>\parbox</code>	34	<code>\underline</code>	30
<code>\pmb</code>	45	<code>\upDelta</code>	59
<code>\poptabs</code>	20	<code>\upOmega</code>	59
<code>\ps</code>	26	<code>\upshape</code>	44
<code>\pushtabs</code>	20	<code>\usebox</code>	35
<code>\put</code>	19	<code>\usecounter</code>	10
<code>\raggedbottom</code>	13	<code>\usefont</code>	45
<code>\raggedleft</code>	16	<code>\usepackage</code>	13
<code>\raggedright</code>	16	<code>\value</code>	11
<code>\raisebox</code>	34	<code>\vdots</code>	30
<code>\ref</code>	11	<code>\vector</code>	19
<code>\refstepcounter</code>	10	<code>\verb</code>	24
<code>\rmfamily</code>	44	<code>\vfill</code>	33
<code>\roman</code>	10	<code>\vline</code>	22
<code>\rule</code>	35	<code>\vpageref</code>	60
<code>\samepage</code>	27	<code>\vpagerefrange</code>	60
<code>\savebox</code>	35	<code>\vref</code>	60
<code>\sbox</code>	35	<code>\vrefrange</code>	60
<code>\scriptsize</code>	44	<code>\vspace</code>	33
<code>\scshape</code>	44	<code>\width</code>	25
<code>\selectfont</code>	45		
<code>\setcounter</code>	10	A	
<code>\setlength</code>	25	About this Document	115
<code>\settodepth</code>	25	Accents	35
<code>\settoheight</code>	25	Accessing any character of a font	35
<code>\settowidth</code>	25	Aligning Equations	15
<code>\sffamily</code>	44	amsmath	45
<code>\shortstack</code>	19	ansinew	50
<code>\signature</code>	27	Appendix, creating	32
<code>\sloppy</code>	28	applemac	50
<code>\slshape</code>	44	array	14
<code>\small</code>	44	Arrays, math	14
<code>\smallskip</code>	33	Arrows	36
<code>\sqrt</code>	30	article class	12
<code>\startbreaks</code>	27	Author, for titlepage	31
<code>\stepcounter</code>	10		
<code>\stopbreaks</code>	27	B	
<code>\subparagraph</code>	32	BCOR	51
<code>\subsection</code>	32	Bibliography, creating (automatically)	23

Bibliography, creating (manually)	22
BibTeX	79
bib files	79
@STRING command	79
Entry Format	80
Entry Types	80
Field Text	84
Field Types	86
bst files	89
ABBRV.BST	89
ALPHA.BST	89
PLAIN.BST	89
SAMPLE.BST	89
UNSRT.BST	90
Command Qualifiers	79
Entry Types	
article entry	81
book entry	81
booklet entry	81
conference entry	81
inbook entry	82
incollection entry	82
inproceedings entry	82
manual entry	82
mastersthesis entry	82
misc entry	83
phdthesis entry	83
proceedings entry	83
techreport entry	83
unpublished entry	83
Field Text	
Abbreviations	85
Names	84
Titles	85
Field Types	
address field	86
annotate field	86
author field	86
booktitle field	86
chapter field	86
edition field	86
editor field	86
howpublished field	87
institution field	87
journal field	87
key field	87
month field	87
note field	87
number field	87
organization field	88
pages field	88
publisher field	88
school field	88
series field	88
title field	88
type field	88
volume field	89
year field	89
Parameters	79
using	23
Binary Operators	37
bm	60
book class	12
Boxes	32

C	
Cc list	26
center	14
Centering text	14
Centering text, environment for	14
Characters, reserved	35
Characters, special	35
cite.sty	23
Classes of document	12
Commands, defining new ones	11
Computer programs, typesetting	24
ConTeXt	71
Counters, a list of	10
Counters, creating	10
Counters, getting the value of	11
Counters, setting	10
cp850	50
Creating letters	25
Creating pictures	17
Creating tables	21
Credits	5
Cross referencing	11
Cross referencing using page number	11
Cross referencing using section number	11
D	
Date, for titlepage	31
dcolumn	46
Defining a new command	11
Defining new environments	11
Defining new fonts	12
Defining new theorems	12
Delimiters	37
description	14
Displaying quoted text	20
Displaying quoted text with paragraph indentation	20
DIV	51
Document Classes	12
E	
Ellipsis	39
Enclosed material	26
Ending & Starting	42
Enlarge current page	27
enumerate	15
Environments	13
Environments, defining	11
eqnarray	15
equation	15
Equations, aligning	15
Equations, environment for	15
exponent	29
F	
figure	15
Figures, footnotes in	17
Flushing a page	27
flushleft	16
flushright	16
Font commands, low-level	44
Font Sizes	44
Font Styles	44
fontenc	46
fontinst	71
Fonts	43
Fonts, new commands for	12

footexclude	51
footinclude	51
Footnotes in figures	17
Footnotes, creating	24
Formatting Text	14
Formulae, environment for	15
Formulae, maths	29

G

german	46
!-	46
"	46
"=	46
"	46
"~	46
"a	46
"ck	46
"ff	46
\-	46
\dq	47
\flq	46
\flqq	46
\frq	46
\frqq	46
\glq	46
\glqq	46
\grqq	46
<	47
Global options	13
graphics	47
Greek Letters	39
Greek letters	29

H

headexclude	51
headinclude	51
hyperref	47
commands	
\autoref	50
\href	49
\hyperbaseurl	49
\hyperimage	50
\hyperlink	50
\hyperref	50
\hypertarget	50
\nolinkurl	50
\texorpdfstring	49
options	
4	47
a4paper	47
a5paper	47
anchorcolor	47
b5paper	47
backref	47
baseurl	47
bookmarks	47
bookmarksnumbered	47
bookmarksopen	47
bookmarksopenlevel	47
bookmarkstype	47
breaklinks	47
citebordercolor	48
citecolor	48
colorlinks	48
debug	48
draft	48

dvipdf	48
dvipdfm	48
dvips	48
dvipsone	48
dviwindo	48
executivepaper	47
extension	48
filebordercolor	48
filecolor	48
frenchlinks	48
hyperfigures	48
hyperindex	48
hypertex	48
hypertexnames	48
implicit	48
latex2html	48
legalpaper	47
letterpaper	47
linkbordercolor	48
linkcolor	48
linktocpage	48
menubordercolor	48
menucolor	48
naturalnames	48
nesting	48
pageanchor	48
pagebackref	48
pagebordercolor	48
pagecolor	48
pdfauthor	48
pdfborder	48
pdfcenterwindow	48
pdfcreator	48
pdfffitwindow	48
pdfhighlight	49
pdfkeywords	49
pdfmenubar	49
pdfnewwindow	49
pdfpagelayout	49
pdfpagemode	49
pdfpagescrop	49
pdfpagetransition	49
pdfproducer	49
pdfstartpage	49
pdfstartview	49
pdfsubject	49
pdftex	48
pdftitle	49
pdftoolbar	49
pdfview	49
pdfwindowui	49
plainpages	49
raiselinks	49
runbordercolor	49
tex4ht	48
textures	48
unicode	49
urlbordercolor	49
urlcolor	49
verbose	49
vtex	48
Hyphenation, defining	28
Hyphenation, forcing	27

I

Indent, forcing	29
-----------------	----

Indent, suppressing 29
 Input file, splitting 42
 Input/Output 43
 inputenc 50
 Inserting figures 15
 itemize 17

J

Justification, ragged left 16
 Justification, ragged right 16

K

Known Issues 115
 KOMA-Script 51
 scraddr 59
 \Address{} 59
 \Comment{} 59
 \FirstName{} 59
 \FreeIII{} 59
 \FreeII{} 59
 \FreeIV{} 59
 \FreeI{} 59
 \InputAddressFile 59
 \LastName{} 59
 \Name{} 59
 \Telephone{} 59
 scrlltr2 51
 \KOMAOptions{} 51
 \addtoeffields 55
 \ifkomavareempty 56
 \newkomavar 55
 \setkomavar 56
 \usekomavar 56
 address 57
 address files 59
 backaddress 55
 backaddressseparator 55
 business line 57
 carbon copy 58
 ccseparator 55
 circular letters 59
 Closing 58
 Closing Part 58
 customer 55
 date 55
 emailseparator 55
 enclosure 58
 encseparator 55
 faxseparator 55
 font options 52
 font selection 54
 footer 57
 footnotes 58
 format options 54
 fromaddress 55
 frombank 55
 fromemail 55
 fromfax 55
 fromlogo 55
 fromname 55
 fromphone 55
 fromurl 55
 general document properties 54
 invoice 55
 language selection 59
 language support 58

letter class option files 54
 letter declaration 56
 letterhead 56
 lists 58
 location 55
 margin notes 58
 myref 55
 Opening 58
 options 51
 options for letterhead and address 52
 other layout 51
 page layout 51
 page style 54
 phoneseparator 55
 place 55
 placeseparator 55
 postscript 58
 pseudo lengths 56
 sender's extension 57
 signature 56
 specialmail 56
 subject 56
 subject line 57
 subjectseparator 56
 text 58
 text emphasis 58
 title 56
 title line 57
 toaddress 56
 toname 56
 variables 55
 yourmail 56
 yourref 56

L

Labelled lists, creating 14
 LaTeX overview 9
 LaTeX Packages 45
 latin1 50
 latin9 50
 Layout commands 13
 Left-justifying text 16
 Left-justifying text, environment for 16
 Left-to-right mode 30
 Lengths, adding to 25
 Lengths, defining a new 25
 Lengths, defining and using 25
 Lengths, predefined 25
 Lengths, setting value of 25
 letter 17
 letter class 12
 Letters 25
 Letters, ending 26
 Letters, starting 26
 Line Breaking 27
 Line breaks 28
 Lines in tables 21
 Lining text up in columns using tab stops 20
 Lining text up in tables 21
 list 17
 Lists of items 17
 Lists of items, generic 17
 Lists of items, numbered 15
 Loading additional packages 13
 Low-level font commands 44
 LR mode 30

lrbox	34
M	
Make-Index	91
Example	98
Input Style Specifiers	93
actual	93
arg_close	93
arg_open	93
encap	93
escape	93
keyword	93
level	93
quote	94
range_close	94
range_open	94
Options	91
-L	92
-T	92
-c	91
-g	91
-i	91
-l	91
-o ind	92
-p no	92
-q	91
-r	92
-s sty	92
-t log	92
Ordering	98
Output Style Specifiers	94
delim_0	96
delim_1	96
delim_2	96
delim_n	96
delim_r	96
delim_t	97
encap_infix	97
encap_prefix	97
encap_suffix	97
group_skip	94
heading_prefix	95
heading_suffix	95
headings_flag	95
indent_length	97
indent_space	97
item_0	95
item_01	96
item_1	95
item_12	96
item_2	95
item_x1	96
item_x2	96
line_max	97
numhead_negative	95
numhead_positive	95
page_compositor	94
postamble	94
preamble	94
setpage_prefix	94
setpage_suffix	94
suffix_2p	97
suffix_3p	97
suffix_mp	98
symhead_negative	95
symhead_positive	95

Special Effects	98
Style File	93
Making a title page	23
Making paragraphs	28
Margin Notes	29
Math Formulae	29
Math Functions	41
Math Miscellany	30
Math mode	30
Math mode, entering	29
Math mode, spacing	30
Math Symbols, Variable Size	41
mathpazo	59
Maths symbols	29
Metafont	75
MetaPost	76
Environment	77
MFINPUTS	77
MPEDIT	78
MPINPUTS	77
MPSUPPORT	77
MPXCOMMAND	77
TEX	77
TROFF	78
Options	76
-file-line-error-style	76
-help	76
-ini	76
-interaction mode	76
-jobname name	76
-kpathsea-debug bitmask	76
-mem mem	76
-parse-first-line	77
-progname name	77
-recorder	77
-translate-file tcxname	77
-troff	77
-version	77
-T	77
minipage	17
Minipage, creating a	17
Modes	30
Multicolumn text	13
N	
New line, starting	27
New line, starting (paragraph mode)	28
New Page	28
Notes in the margin	29
O	
Operators	
Binary	37
Relational	37
Options, global	13
overcite.sty	23
Overview of LaTeX	9
P	
Packages	
amsmath	45
bm	60
bm.sty	60
dcolumn	46
german	46
graphics	47
hyperref	47

KOMA-Script	51	Programs, typesetting	24
mathpazo	59	PSTricks	61
Several Small Packages	60	commands	
url	60	\Aput	68
varioref	60	\Bput	68
Packages, loading	13	\NormalCoor	69
Page break, forcing	28	\PSTtoEPS	70
Page Breaking	27	\Rnode	67
Page Formatting	27	\RnodeRef	67
Page numbering	31	\SpecialCoor	69
Page styles	30	\TeXtoEPS	70
Paragraph mode	30	\aput	68
Paragraph, starting a new	29	\arrows	66
Paragraphs	28	\bput	68
Parameters	45	\circlenode	67
pdfTeX	72	\clipbox	67
Environment	73	\closedshadow	65
TEXEDIT	74	\closepath	65
TEXFONTS	74	\cnode	67
TEXFORMATS	74	\cnodeput	67
TEXINPUTS	74	\code	65
TEXMFOUTPUT	74	\coor	65
TEXPOOL	74	\cput	67
Options	72	\curveto	65
-file-line-error-style	72	\dataplot	63
-fmt format	72	\degrees	61
-help	72	\dim	65
-ini	72	\endpspicture	66
-interaction mode	72	\everypsbox	70
-ipc	72	\file	66
-ipc-start	72	\fileplot	63
-jobname name	72	\fill	65
-kpathsea-debug bitmask	73	\grestore	65
-maketex fmt	73	\gsave	65
-no-maketex fmt	73	\lineto	65
-output-comment string	73	\listplot	63
-parse-first-line	73	\lput	68
-progname name	73	\movepath	65
-recorder	73	\moveto	65
-shell-escape	73	\mput	68
-translate-file tcxname	73	\mrestore	65
-version	73	\msave	65
People		\multips	66
Bausum, David	9	\multirput	66
Berry, Karl	5	\ncLine	68
Borgert, Wolfgang	5	\ncangle	68
Caseelas, Ramon	5	\ncangles	68
Devenish, James	5	\ncarc	68
Gaesslein, Hubert	5	\ncbar	68
Goossens, Michel	9	\ncircle	68
Hagen, Hans	71	\nccoil	69
Karp, Peter	5	\nccurve	68
Lamport, Leslie	9	\ncdiag	68
Mittelbach, Frank	9	\ncdiagg	68
Niepraschk, Rolf	5	\ncline	68
Pepping, Simon	5	\ncloop	68
Samarin, Alexander	9	\nczigzag	69
Stayton, Bob	5	\newcmkcolor	61
van Zandt, Timothy	61	\newgray	61
Voss, Herbert	5	\newhsbcolor	61
PiCTeX	60	\newpath	65
picture	17	\newpsobject	64
Pictures, creating	17	\newpsstyle	64
PlainTeX	9	\newrgbcolor	61
Poetry, an environment for	24	\openshadow	65
Predefined lengths	25	\ovalnode	67

<code>\overlaybox</code>	69	<code>\rotateleft</code>	67
<code>\parabola</code>	62	<code>\rotateright</code>	67
<code>\parametricplot</code>	63	<code>\rput</code>	66
<code>\pangle</code>	68	<code>\savedata</code>	63
<code>\pcarc</code>	68	<code>\scale</code>	65
<code>\pchar</code>	68	<code>\scalebox</code>	67
<code>\pcoil</code>	69	<code>\scaleboxto</code>	67
<code>\pccurve</code>	68	<code>\setcolor</code>	66
<code>\pcdiag</code>	68	<code>\stroke</code>	65
<code>\pcline</code>	68	<code>\swapaxes</code>	65
<code>\pclip</code>	68	<code>\translate</code>	65
<code>\pcoil</code>	69	<code>\uput</code>	66
<code>\pnode</code>	67	parameter	
<code>\psCoil</code>	69	angle	68
<code>\psarc</code>	62	arcangle	68
<code>\psarcn</code>	62	arcsep	62
<code>\psaxes</code>	66	arcsepA	62
<code>\psbezier</code>	62	arcsepB	62
<code>\psccurve</code>	62	arm	67
<code>\pscharclip</code>	70	arrowinset	64
<code>\pscharpath</code>	70	arrowlength	64
<code>\pscircle</code>	62	arrows	64
<code>\pscirclebox</code>	67	arrowscale	64
<code>\psclip</code>	67	arrowsize	64
<code>\pscoil</code>	69	axesstyle	66
<code>\pscurve</code>	62	bblx	70
<code>\pscustom</code>	64	bbly	70
<code>\psdblframebox</code>	67	bburx	70
<code>\psdots</code>	62	bbury	70
<code>\psecurve</code>	62	border	63
<code>\psellipse</code>	62	bordercolor	63
<code>\psframe</code>	61	boxsep	66
<code>\psframebox</code>	67	bracketlength	64
<code>\psgrid</code>	62	coilheight	69
<code>\pslabelsep</code>	66	coilinc	69
<code>\pslbrace</code>	70	coilwidth	69
<code>\psline</code>	61	cornersize	61
<code>\pslongbox</code>	70	curvature	62
<code>\psmathboxfalse</code>	70	dash	63
<code>\psmathboxtrue</code>	70	dimen	64
<code>\psovalbox</code>	67	dotangle	62
<code>\psoverlay</code>	69	dotscale	62
<code>\pspicture</code>	66	dotsep	63
<code>\psplot</code>	63	dotsize	64
<code>\pspolygon</code>	61	dotstyle	62
<code>\psrbrace</code>	70	doublecolor	63
<code>\psset</code>	61	doubleline	63
<code>\pssetlength</code>	61	doublesep	63
<code>\psshadowbox</code>	67	fillcolor	64
<code>\pstextpath</code>	70	fillstyle	64
<code>\psverbboxfalse</code>	70	framearc	61
<code>\psverbboxtrue</code>	70	framesep	66
<code>\pswedge</code>	62	gradangle	70
<code>\psxlabel</code>	66	gradbegin	69
<code>\psylabel</code>	66	gradend	69
<code>\pszigzag</code>	69	gradlines	69
<code>\putoverlaybox</code>	69	gradmidpoint	70
<code>\qline</code>	61	gridcolor	62
<code>\radians</code>	61	griddots	62
<code>\rcoor</code>	65	gridlabelcolor	62, 63
<code>\rcurveto</code>	65	gridlabels	62
<code>\readdata</code>	63	gridwidth	62
<code>\rlineto</code>	65	hatchangle	64
<code>\rnode</code>	67	hatchcolor	64
<code>\rotate</code>	65	hatchsep	64
<code>\rotatedown</code>	67	hatchwidth	64

- | | | | |
|---|----|-------------------------------|-----|
| headerfile | 70 | Starting on a right-hand page | 27 |
| headers | 70 | Styles of text | 44 |
| labels | 66 | Styles, page | 30 |
| labelsep | 66 | Subscript | 29 |
| liftpen | 65 | Superscript | 29 |
| linearc | 61 | Symbols | 29 |
| linecolor | 61 | Miscellaneous | 40 |
| linestyle | 63 | Special | 35 |
| linetype | 64 | T | |
| linewidth | 61 | Tab stops, using | 20 |
| loopsize | 68 | tabbing | 20 |
| ncurv | 68 | table | 21 |
| nodesep | 67 | Table of Contents, creating | 43 |
| offset | 67 | Tables, creating | 21 |
| origin | 63 | tabular | 21 |
| plotpoints | 63 | Terminal Input/Output | 43 |
| plotstyle | 63 | Texinfo | 71 |
| rbracketlength | 64 | Thanks, for titlepage | 31 |
| runit | 61 | thebibliography | 22 |
| shadow | 63 | theorem | 23 |
| shadowangle | 63 | Theorems, defining | 12 |
| shadowcolor | 64 | Theorems, typesetting | 23 |
| shadowsize | 63 | Title making | 31 |
| showorigin | 66 | Title pages, creating | 23 |
| showpoints | 61 | Title, for titlepage | 31 |
| subgriddots | 63 | titlepage | 23 |
| subgridwidth | 63 | Typed text, simulating | 24 |
| swapaxes | 63 | Typeface Sizes | 44 |
| tbarsize | 64 | Typeface Styles | 44 |
| ticks | 66 | Typefaces | 43 |
| ticksize | 66 | U | |
| tickstyle | 66 | url | 60 |
| unit | 61 | Using BibTeX | 23 |
| xunit | 61 | V | |
| yunit | 61 | Variables, a list of | 10 |
| Q | | varioref | 60 |
| quotation | 20 | verbatim | 24 |
| quote | 20 | Verbatim text | 24 |
| Quoted text with paragraph indentation, | | verse | 24 |
| displaying | 20 | Vertical space, inserting | 33 |
| Quoted text, displaying | 20 | X | |
| R | | xindy | 101 |
| Ragged left text | 16 | Command List | 101 |
| Ragged left text, environment for | 16 | Markup Commands | 107 |
| Ragged right text | 16 | Raw Index Interface | 112 |
| Ragged right text, environment for | 16 | Invoking | 113 |
| Relational Operators | 37 | Command Line Options | 113 |
| Remarks in the margin | 29 | Search Path | 114 |
| report class | 12 | Processing Commands | 101 |
| Reserved Characters | 35 | define-alphabet | 101 |
| Right-justifying text | 16 | define-attributes | 101 |
| Right-justifying text, environment for | 16 | define-crossref-class | 102 |
| S | | define-letter-group | 102 |
| Sectioning | 32 | define-location-class | 103 |
| Several Small Packages | 60 | define-location-class-order | 103 |
| Simulating typed text | 24 | define-rule-set | 103 |
| Sizes of text | 44 | define-sort-rule-orientations | 104 |
| Small Packages | 60 | merge-rule | 104 |
| Space, inserting vertical | 33 | merge-to | 105 |
| Spaces | 32 | require | 105 |
| Spacing, within Math mode | 30 | searchpath | 106 |
| Special Characters | 35 | sort-rule | 106 |
| Splitting the input file | 42 | use-rule-set | 106 |
| Starting & Ending | 42 | | |