A. Syropoulos, A. Tsolomitis and N. Sofroniou,
Digital Typography Using $\LaTeX$
10 October 2002
ISBN 0-387-95217-9

Errata

What follows is a list of pages from the book

“Digital Typography Using LaTeX”
by
A. Syropoulos, A. Tsolomitis and N. Sofroniou,

which contain errors which appeared during the production of the book.
The great majority of them relates to the minus sign ($-$) which disappeared during printing. This affected mainly Chapter 5 which deals with Mathematics. It also affected long arrows that use this character to extend their length like $\longrightarrow$ (`\longrightarrow`) etc.
Errata that do not relate to the above problem are very few (2 or 3 symbols like $\boxdot$ (`\boxdot`)).
All pages with errata are reproduced in the next pages exactly as they appear in the book but the errors are corrected and are put in a gray background.

the first enthusiastic years of explosion of the Web, people realized that HTML (even combined with CSS) was definitely *not* sufficient for formatting documents. XML provided the necessary standard for structuring documents in an arbitrarily fine way, but still there was no "standard" way to *represent* an XML document. In October 2001, a new standard filled that gap: XSL-FO. The tools provided by XSL-FO for formatting documents are a quite serious challenge, and a new generation of XSL-FO-compliant typesetting engines is slowly emerging.

More generally, the current trend is to use XML as the basis of every kind of file format. For example, the SVG standard is, in some sense, an "XML-ized version of PostScript." One could very well imagine all file formats involved in TEX becoming XML-compliant: the input file could be pure XML "processing instructions" for including code in the TEX language the DVI file format could be replaced by SVG, the font metrics could be expressed in XML, illustrations could be in SVG instead of EPS, and so on. In that case, TEX (or Ω, or some other successor to TEX) would simply transform one XML document into another one. The fact that XML document transformation is nowadays an increasingly popular and important concept is by no means a coincidence.

Another area where Ω can be applied to revolutionize the electronic document is that of adaptive documents. A research project in that area deals with *vario-documents*, namely documents that contain a big number of page descriptions and display the right one according to context parameters, just as HTML browsers reflow text when their display window is resized. Only here each page description of the document has been compiled in advance by a "super-Ω," always with the same high typesetting quality standards.

Yet another area of drastic improvement of Ω's capabilities would be an on-the-fly interaction between typesetting and dynamic fonts. Already, in VectorTEX (a commercial TEX for Windows platform), Dimitri Vulis has included METAFONT capabilities into TEX. By using more modern font formats, such as OpenType, one could obtain a dialog between the font and TEX's typesetting engine so that each one instructs the other on constraints and context parameters and so that the final result is optimal for both.

There is also the more global, operating system-oriented point of view: Ω could very well become a server, and arbitrary client applications could send requests with text extracts and macros or parameters and receive in return small parts of page descriptions.

All of these "mutation" scenarios could be compared with the common skeleton of many science-fiction stories, where humans mutate to become less and less organic. Usually sci-fi authors want to express the fact that despite and beyond the changes of the human body (including an artificial brain), a core of *humanity* will always emerge as a fundamental quality of mankind. This is exactly the case for TEX: I am convinced that however drastically TEX (and its successors) will change in the future, its fundamental quality, which is the love of one man—and not just any man!—for good typography and good programming will always prevail and will always be the ultimate guarantee for the survival of this magnificent tool.

```
                                 \ifx\firstarg\empty
                                    \relax ...
                                 \else ... \fi}
      \newcommand{\cmdb}[1]{...}
```

The `\ifx` construct is used to check whether the first argument is empty or not. In case it is, we put the code that processes the second argument after the command `\relax`, otherwise we put the code that processes both arguments after the command `\else`. The `\relax` command prevents TeX from consuming more tokens than is necessary, and it actually does nothing.

In case one wants to change the appearance of the sectioning commands, the command can be redefined `\@seccntformat` can be redefined . For example, if we would like old style numerals, then the following redefinition achieves the desired effect:

```
\renewcommand{\@seccntformat[1]}{%
    \oldstylenums{\csname the#1\endcsname}\quad}
```

The command `\oldstylenums` should be used to typeset a number using old style numerals. The construct `\csname` *string* `\endcsname` makes up a command of the *string*. To turn a command into a string, use the `\string` command.

The table of contents of a document can be constructed by issuing the command `\tableofcontents`. This command can be put in any place that the user feels is appropriate. However, it is common practice to have the table of contents in either the beginning of the document, usually after the title or title page, or at the end, just before the index and the bibliography.

In some cases, one may want to be able to manually add something to the table of contents. In cases such as this, one can use the command

```
\addcontentsline{table}{type}{entry}
```

where *table* is the file that contains the table of contents, *type* is the type of the sectioning unit, and *entry* is the actual text that will be written to the table of contents. For example, the command

```
\addcontentsline{toc}{chapter}{Preface}
```

adds to the `.toc` file (i.e., the table of contents) the chapter entry `Preface`. In case we want to add an entry to the list of tables or figures, the `table` is called `lot` or `lof`, respectively (see Section 6.5).

The command `\addcontentsline` is defined in terms of the commands `\addcontents` and `\contentsline`. The first command has two arguments: the *table* and a *text*; this command adds the *text* to the *table* file, with no page number. The second command has three arguments: the *type*, the *entry*, and the *page*, which can be either a number or a command that yields the current page number; the command produces a *type* entry in the *table*. For example, the entry for Section 2.5 of this book in the table of contents is produced by the command

```
\contentsline{section}{\numberline{2.5}Basic Logos}{26}
```

The command `\numberline` puts its argument flush left in a box whose width is stored in the internal length variable `\@tempdima`. Now, we give the definition of `\addcontentsline`:

```
\newcommand\addcontentsline[3]{%
  \addtocontents{#1}{%
  \protect\contentsline{#2}{#3}{\thepage}}}
```

➤ **Exercise 2.6** Create a new sectioning command that will behave like the `\section` command and in addition will put an asterisk in front of the section number in the table of contents. [Hint: Use the `\let` command (see Section 4.5.1)] □

Every document class defines for each sectioning command an `\l@`*type* command. The general form of this command is

`\l@`*type*`{`*entry*`}{`*page*`}`

These commands are needed for making an *entry* of type *type* in a table of contents, or elsewhere. Most of the `\l@`*type* commands are applications of the command

`\@dottedtocline{`*level*`}{`*indent*`}{`*numwidth*`}{`*title*`}{`*page*`}`

where *indent* is the total indentation from the left margin, *numwidth* the width of the box that contains the section number if the *title* (i.e., the contents of the entry) has a `\numberline` command, and *page* the page number. Here is an example:

```
\newcommand*\l@section{\@dottedtocline{1}{1.5em}{2.6em}}
```

Note that the `\@dottedtocline` produces a dotted line. If we do not like this effect, then a good solution is to try to delete the part that produces the leaders.

When creating a large document, one may need to have appendices. LaTeX provides the command `\appendix`, which resets the numbering of the various sectioning commands and, depending on the document class, forces the top sectioning command to produce alphabetic numbers instead of Arabic numbers. So, if we are preparing a book, the chapter numbers in the appendix appear as letters.

If we are preparing a book, we usually want to have a preface and/or a prologue. Moreover, it is customary for the page numbers of this part of our document to be typeset using the Roman numbering system (i.e., i, ii, iii, etc.). The `book` document class provides the commands `\frontmatter`, `\mainmatter`, and `\backmatter`, which can be used to divide a book into three (logical) sections. The effect of the first command is to start Roman page numbering and to turn off chapter numbering. The second

Note that a book is not allowed to have an abstract, as this makes no sense.

➤ **Exercise 2.7** Write a small document and write the abstract before and after the command `\maketitle`. What's the difference? □

Since what LaTeX provides to its users may not satisfy everybody, the system allows its users to reprogram the `\maketitle` command and its associated commands or just use the `titlepage`. This environment generates a new page where the author is free to put whatever suits the needs of the particular document being prepared. Of course, at this moment, our knowledge of LaTeX is too limited so there are only a few things we can do. As the reader proceeds with this the book, new commands to create an excellent title page will be discovered.

2.5 Basic Logos

A logo is a name, symbol, or trademark designed for easy and definite recognition, so the word TeX is a logo. The natural question is: ''How can one produce this and the other TeX-related logos?'' The answer is in the following table.

Logo	Command
TeX	`\TeX`
LaTeX	`\LaTeX`
LaTeX 2ε	`\LaTeXe`

All of these commands have a peculiar behavior, which is demonstrated by the following example (note that the character ␣ is a visual representation of the space character):

Plain TeXis easy	`Plain␣\TeX␣␣␣␣␣is␣easy`
but LaTeX is easier!	`but␣\LaTeX\␣is␣easier!`

It is obvious that in the first example, regardless of the number of spaces that follow the command `\TeX`, TeX is setting no space between the logo and the next word. This happens simply because these commands *consume* all of the white space that follows them. However, in the second example, we see that the use of the command `\␣` produces the intended result. The effect of this command is to force TeX to produce a reasonable interword space.

➤ **Exercise 2.8** According to the regulations set by the Greek Postal Services, when one affixes a printed address label on a postal object, the first three digits of the postal code must be separated by a single space from the remaining two digits. Moreover, we have to add two spaces between the postal code and the name of the town that immediately follows. Write down the necessary commands to typeset the following address according to the regulations of the Greek Postal Services:

Astrological symbols and the zodiacal symbols:

`\aries`	♈	`\taurus`	♉	`\gemini`	♊	`\cancer`	♋
`\leo`	♌	`\virgo`	♍	`\libra`	♎	`\scorpio`	♏
`\sagittarius`	♐	`\capricornus`	♑	`\aquarius`	♒	`\pisces`	♓
`\conjunction`	☌	`\opposition`	☍				

Musical notes:

`\eighthnote`	♪	`\fullnote`	𝅝	`\halfnote`	𝅗𝅥	`\quarternote`	♩
`\twonotes`	♫						

APL symbols (i.e, symbols used in APL programs):

`\APLbox`	⎕	`\APLcirc{*}`	⊛	`\APLcomment`	⍝
`\APLdown`	∇	`\APLdownarrowbox`	⍗	`\APLinput`	⍞
`\APLinv`	⌹	`\APLleftarrowbox`	⍇	`\APLlog`	⍟
`\APLminus`	‾	`\APLnot{*}`	⍨	`\APLrightarrowbox`	⍈
`\APLstar`	⋆	`\APLup`	∆	`\APLuparrowbox`	⍐
`\APLvert{*}`	⍿	`\notbackslash`	⍀	`\notslash`	⌿

Note that `\APLcirc`, `\APLnot`, and `\APLvert` have a required argument.
Polygons and stars:

`\CheckedBox`	☑	`\davidsstar`	✡	`\hexagon`	⎔	`\hexstar`	✳
`\octagon`	⯃	`\pentagon`	⬠	`\Square`	□	`\varhexagon`	⬡
`\varhexstar`	✻	`\XBox`	☒				

3.2.3 Phonetic Fonts

For linguistic purposes, a special phonetic font is available using the package phonetic by Emma Pease. Table 3.7 gives the commands provided by the package. Note that the oblique glyphs can be accessed by feeding the corresponding command as argument to the `\textit` command. The following example shows the phonetic transcriptions of Navaho (native USA) words:

dlő́ő́ʔ	(prairie dog)	tʃʼah	(hat)
łitsx^{w}o	(yellow-orange)	tsı̋ı̋ł	(haste)
ʔakʼos	(neck)	xaɩh	(winter)

People interested in linguistics and TeX may find more information at the following URL: `http://www.ifi.uio.no/~dag/ling-tex.html`.

fitness *ﬁtness*
fitness ﬁtness

Figure 3.5: Fitness without and with ligatures.

provides the commands `\textendash` and `\textemdash`, which produce an en dash or an em dash, respectively.

The plain dash, or hyphen, is the ''joining'' dash as in ''David Ben-Gurion Airport.'' The en dash is used for ranges: ''the information is on pages 17–58'' and to join two proper names such as ''Heine–Borel'' to avoid confusion with single persons with a hyphenated surname (e.g., Ben-Gurion). The em dash is the dash used for parenthetical purposes: ''she was angry —please believe me— and this is why she did not talk to us.''

Ligatures have another use also. Some languages (or for ornamental reasons in Latin-derived languages) require many accents or letters to change form depending on their location in the text. This is very common in Arabic, for example. Polytonic writing is needed to typeset classical Greek texts. For example, we write `θα>ρ<ρ˜ων`, and the ligature mechanism combines the special characters with the following letter to produce ϑαῤῥῶν. The following text is from Μονῳδια επι Ιουλιανῳ of Λιβανιος.

> Ὢ πόποι, ἦ μέγα πένθος οὐκ᾿ Ἀχαιίδα γῆν μόνον, ἀλλὰ καὶ πᾶσαν ὁπόσην ὁ ῾Ρωμαίων κοσμεῖ ϑεσμός, κατείληφε· μᾶλλον μὲν γὰρ ἴσως ἦν Ἕλληνες οἰκοῦσιν, ἅτε και μᾶλλον αἰσθανομένην τοῦ κακοῦ, διήκει δ᾿ οὖν καὶ διὰ πάσης γῆς, ὥς ἔφην, ἡ πληγὴ τύπτουσά τε καὶ κατατέμνουσα τὰς ψυχάς, ὥς οὐκέτ᾿ ὂν βιωτὸν ἀνδρὶ βελτίστῳ τε καὶ ὅτῳ τοῦ εὖ ζῆν ἐπιθυμία.

To get this text, we have typed

```
>‘Ω πόποι, >˜η μέγα πένθος ο>υκ’’ >Αχαιίδα γ˜ην μόνον,
>αλλ‘α κα‘ι π˜ασαν <οπόσην <ο <Ρωμαίων κοσμε˜ι θεσμός,
κατείληφε; μ˜αλλον μ‘εν γ‘αρ >ίσως <‘ην <΄Ελληνες ο>ι-
κο˜υσιν, <άτε και μ˜αλλον α>ισθανομένην το˜υ κακο˜υ,
διήκει δ’’ ο>˜υν κα‘ι δι‘α πάσης γ˜ης, <ως >έφην, <η
πληγ‘η τύπτουσά τε κα‘ι κατατέμνουσα τ‘ας ψυχάς, <ως
ο>υκέτ’’ >‘ον βιωτ‘ον >ανδρ‘ι βελτίστω| τε κα‘ι <ότω|
το˜υ ε>˜υ ζ˜ην >επιθυμία.
```

Many commercial font families provide special letters for the English alphabet for ending or beginning a word. This practice used to be very common with Greek, and it is a necessity for some other languages. That is why D.E. Knuth created the ''vargreek'' letters such as ϑ, θ, φ, ϕ, and so on, but he made them available only in math mode

5 Typesetting Mathematics

One of the most demanding jobs in the typesetting business is the typesetting of mathematical text. Here, TEX really excels. Its output is incomparable to the output of any other document preparation system. LATEX, as usual, adds an easier interface to the TEX typesetting engine, making the writing of even the most demanding mathematical text straightforward.

The American Mathematical Society has made an enormous effort to create LATEX document classes and packages that deal even better with several issues compared to the usual `article` and `book` document classes. They have created the `amsart` and `amsbook` classes, and packages that provide the basic functionality of these classes and work with any document class. The advantages are really important for mathematicians or others who use mathematics in their work, so we thoroughly discuss these classes and packages.

5.1 The Mathematics Mode

When we want to write mathematical text, we must inform LATEX of our intentions. This is done by surrounding the mathematical text with the dollar character: `$`. LATEX takes special care about spacing in math mode even if we write something simple, leading to professional mathematical typesetting. Take a look at the following:

10/5=2 and 7-3=4

$10/5 = 2$ and $7 - 3 = 4$.

The second line looks much better, and it was produced by the input

```
$10/5=2$ and $7-3=4$.
```

When we talk about the "mathematics mode," we must bear in mind that TEX typesets in six different modes. A "mode" is the way TEX reads its input text. There are actually two mathematical modes. We have already shown the first of these—that is, text surrounded by single dollar characters. The other mode is when we want to typeset mathematics in

Note that the commands `\boldmath` and `\unboldmath` are shorthands for the `bold` and the `normal` `\mathversions`.

The selection of sans serif and typewriter fonts in math mode is done with the commands `\mathsf` and `\mathtt`, respectively. Thus,

```
$\mathsf{A}^i_{\mathsf{j}} =\mathtt{W}(\mathtt{\alpha})$
```

will be set as $\mathsf{A}^i_{\mathsf{j}} = \mathtt{W}(\alpha)$. The commands `\textrm`, `\textit`, and so forth work fine in math mode, and they observe spaces. They also observe language, so

```
$x^2=-1 \textbf{ and thus } x=\pm i$
$x^2=-1 \textrm{ \textgreek{>'ara} } x=\pm i$
$f_n \stackrel{\textrm{\textgreek{oμ}}}{\longrightarrow} f$.
```

produces

$$x^2 = -1 \textbf{ and thus } x = \pm i$$
$$x^2 = -1 \text{ ἄρα } x = \pm i$$
$$f_n \stackrel{\text{ομ}}{\longrightarrow} f.$$

We should also add here that if the family has been chosen, the commands `\mathrm`, `\textrm`, and so on, will observe this. Therefore,

```
\sffamily We get $x^2=-1 \textbf{ and thus } x=\pm i$
```

will be set as

We get $x^2 = -1$ **and thus** $x = \pm i$

that is, the ''and thus'' phrase was set in sans serif family and bold series.

5.3 Symbols for the Mathematics Mode

One of the most difficult parts of typesetting mathematics is the wealth of mathematical symbols needed even for simple texts. By ''symbols'' we mean glyphs such as $\forall$, $\exists$, $\rightarrow$, and so on. In addition to symbols, mathematicians need different alphabets, as it is customary to use different fonts for certain tasks such as script capitals $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$ or Greek letters $\alpha, \delta, \epsilon, \varepsilon$.

5.3.1 Special Latin Alphabets

In Section 5.2, we saw a calligraphic alphabet provided by the `\mathcal` command. Sometimes, though, one needs an even more scripted font. Such a font is provided by the mathrsfs package (by Jörg Knappen). The package provides the command `\mathscr`, which gives access to script capitals. For example, `$\mathscr{MATHEMATICS}$` produces $\mathscr{MATHEMATICS}$. A similar option is to use the eucal package of the $\mathcal{AMS}$. If

Table 5.3: Accents in math mode.

$\hat{a}$	`\hat{a}`	$\acute{a}$	`\acute{a}`	$\bar{a}$	`\bar{a}`	$\dot{a}$	`\dot{a}`
$\breve{a}$	`\breve{a}`	$\check{a}$	`\check{a}`	$\grave{a}$	`\grave{a}`	$\vec{a}$	`\vec{a}`
$\ddot{a}$	`\ddot{a}`	$\tilde{a}$	`\tilde{a}`	$\mathring{a}$	`\mathring{a}`		

5.3.4 Binary Operators

Binary operators are symbols such as $+$ or $-$ that are used between two *operands*, which, in turn, are numbers, letters, or formulas. The predefined binary operators that LaTeX provides are shown in Table 5.4. Note that the commands marked with $^{(*)}$ are only available if we use the latexsym package. In particular, one can get the symbols •, · and $*$ in ordinary text mode by using the commands `\textbullet`, `\textperiodcentered`, and `\textasteriskcentered`, respectively.

Additional binary operators are provided by the packages amssymb and amsfonts. These are given in Table 5.5.

Table 5.4: Binary operators.

$\pm$	`\pm`	$\cap$	`\cap`	$\diamond$	`\diamond`	$\oplus$	`\oplus`
$\mp$	`\mp`	$\cup$	`\cup`	$\bigtriangleup$	`\bigtriangleup`	$\ominus$	`\ominus`
$\times$	`\times`	$\uplus$	`\uplus`	$\bigtriangledown$	`\bigtriangledown`	$\otimes$	`\otimes`
$\div$	`\div`	$\sqcap$	`\sqcap`	$\triangleleft$	`\triangleleft`	$\oslash$	`\oslash`
$\ast$	`\ast`	$\sqcup$	`\sqcup`	$\triangleright$	`\triangleright`	$\odot$	`\odot`
$\star$	`\star`	$\vee$	`\vee`	$\lhd$	`\lhd`$^{(*)}$	$\bigcirc$	`\bigcirc`
$\circ$	`\circ`	$\wedge$	`\wedge`	$\rhd$	`\rhd`$^{(*)}$	$\dagger$	`\dagger`
$\bullet$	`\bullet`	$\setminus$	`\setminus`	$\unlhd$	`\unlhd`$^{(*)}$	$\ddagger$	`\ddagger`
$\cdot$	`\cdot`	$\wr$	`\wr`	$\unrhd$	`\unrhd`$^{(*)}$	$\amalg$	`\amalg`

Table 5.5: $\mathcal{AMS}$ binary operators (amssymb package).

$\dotplus$	`\dotplus`	$\smallsetminus$	`\smallsetminus`	$\Cap$	`\Cap`
$\Cup$	`\Cup`	$\barwedge$	`\barwedge`	$\veebar$	`\veebar`
$\doublebarwedge$	`\doublebarwedge`	$\boxminus$	`\boxminus`	$\boxtimes$	`\boxtimes`
$\boxdot$	`\boxdot`	$\boxplus$	`\boxplus`	$\divideontimes$	`\divideontimes`
$\ltimes$	`\ltimes`	$\rtimes$	`\rtimes`	$\leftthreetimes$	`\leftthreetimes`
$\rightthreetimes$	`\rightthreetimes`	$\curlywedge$	`\curlywedge`	$\curlyvee$	`\curlyvee`
$\circleddash$	`\circleddash`	$\circledast$	`\circledast`	$\circledcirc$	`\circledcirc`
$\centerdot$	`\centerdot`	$\intercal$	`\intercal`		

Table 5.9: Large delimiters.

$\rmoustache$	`\rmoustache`	$\lmoustache$	`\lmoustache`	$\rgroup$	`\rgroup`	$\lgroup$	`\lgroup`
$\arrowvert$	`\arrowvert`	$\Arrowvert$	`\Arrowvert`	$\bracevert$	`\bracevert`		

Table 5.10: Arrow symbols (starred symbols are available with the latexsym package).

$\leftarrow$	`\leftarrow`	$\longleftarrow$	`\longleftarrow`	$\uparrow$	`\uparrow`
$\Leftarrow$	`\Leftarrow`	$\Longleftarrow$	`\Longleftarrow`	$\Uparrow$	`\Uparrow`
$\rightarrow$	`\rightarrow` or `\to`	$\longrightarrow$	`\longrightarrow`	$\downarrow$	`\downarrow`
$\Rightarrow$	`\Rightarrow`	$\Longrightarrow$	`\Longrightarrow`	$\Downarrow$	`\Downarrow`
$\leftrightarrow$	`\leftrightarrow`	$\longleftrightarrow$	`\longleftrightarrow`	$\updownarrow$	`\updownarrow`
$\Leftrightarrow$	`\Leftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`	$\Updownarrow$	`\Updownarrow`
$\mapsto$	`\mapsto`	$\longmapsto$	`\longmapsto`	$\nearrow$	`\nearrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`	$\searrow$	`\searrow`
$\leftharpoonup$	`\leftharpoonup`	$\rightharpoonup$	`\rightharpoonup`	$\swarrow$	`\swarrow`
$\leftharpoondown$	`\leftharpoondown`	$\rightharpoondown$	`\rightharpoondown`	$\nwarrow$	`\nwarrow`
$\leadsto^{(*)}$	`\leadsto`				

Table 5.11: $\mathcal{AMS}$ arrows (amssymb package).

$\dashrightarrow$	`\dashrightarrow`	$\dashleftarrow$	`\dashleftarrow`	$\leftleftarrows$	`\leftleftarrows`
$\leftrightarrows$	`\leftrightarrows`	$\Lleftarrow$	`\Lleftarrow`	$\twoheadleftarrow$	`\twoheadleftarrow`
$\leftarrowtail$	`\leftarrowtail`	$\looparrowleft$	`\looparrowleft`	$\leftrightharpoons$	`\leftrightharpoons`
$\curvearrowleft$	`\curvearrowleft`	$\circlearrowleft$	`\circlearrowleft`	$\Lsh$	`\Lsh`
$\upuparrows$	`\upuparrows`	$\upharpoonleft$	`\upharpoonleft`	$\downharpoonleft$	`\downharpoonleft`
$\multimap$	`\multimap`	$\leftrightsquigarrow$	`\leftrightsquigarrow`	$\looparrowright$	`\looparrowright`
$\rightleftarrows$	`\rightleftarrows`	$\rightrightarrows$	`\rightrightarrows`	$\circlearrowright$	`\circlearrowright`
$\twoheadrightarrow$	`\twoheadrightarrow`	$\curvearrowright$	`\curvearrowright`	$\upharpoonright$	`\upharpoonright`
$\rightleftharpoons$	`\rightleftharpoons`	$\downdownarrows$	`\downdownarrows`	$\Rrightarrow$	`\Rrightarrow`
$\Rsh$	`\Rsh`	$\rightsquigarrow$	`\rightsquigarrow`		
$\downharpoonright$	`\downharpoonright`	$\rightarrowtail$	`\rightarrowtail`		

Table 5.12: $\mathcal{AMS}$ negated arrows (amssymb package).

$\nleftarrow$	`\nleftarrow`	$\nrightarrow$	`\nrightarrow`	$\nLeftarrow$	`\nLeftarrow`
$\nRightarrow$	`\nRightarrow`	$\nleftrightarrow$	`\nleftrightarrow`	$\nLeftrightarrow$	`\nLeftrightarrow`

5.3.8 Relational Operators

Again, there are two sets of such operators: the standard ones (Table 5.13) and the ones provided by the amssymb package (Table 5.14).

The negated form of a relational operator (i.e., $\not\leq$) is obtained by prepending the command `\not` to the relational operator. For instance, the code `x\not\leq y` is being typeset as $x \not\leq y$. However, the $\mathcal{AMS}$ provides special fonts and commands to access negated relational operators, which are shown in Table 5.15.

Table 5.15: $\mathcal{AMS}$ negated relational operators (amssymb package).

$\nless$	`\nless`	$\nleq$	`\nleq`	$\nleqslant$	`\nleqslant`
$\nleqq$	`\nleqq`	$\lneq$	`\lneq`	$\lneqq$	`\lneqq`
$\lvertneqq$	`\lvertneqq`	$\lnsim$	`\lnsim`	$\lnapprox$	`\lnapprox`
$\nprec$	`\nprec`	$\npreceq$	`\npreceq`	$\precnsim$	`\precnsim`
$\precnapprox$	`\precnapprox`	$\nsim$	`\nsim`	$\nshortmid$	`\nshortmid`
$\nmid$	`\nmid`	$\nvdash$	`\nvdash`	$\nvDash$	`\nvDash`
$\ntriangleleft$	`\ntriangleleft`	$\ntrianglelefteq$	`\ntrianglelefteq`	$\nsubseteq$	`\nsubseteq`
$\subsetneq$	`\subsetneq`	$\varsubsetneq$	`\varsubsetneq`	$\subsetneqq$	`\subsetneqq`
$\varsubsetneqq$	`\varsubsetneqq`	$\ngtr$	`\ngtr`	$\ngeq$	`\ngeq`
$\ngeqslant$	`\ngeqslant`	$\ngeqq$	`\ngeqq`	$\gneq$	`\gneq`
$\gneqq$	`\gneqq`	$\gvertneqq$	`\gvertneqq`	$\gnsim$	`\gnsim`
$\gnapprox$	`\gnapprox`	$\nsucc$	`\nsucc`	$\nsucceq$	`\nsucceq`
$\succnsim$	`\succnsim`	$\succnapprox$	`\succnapprox`	$\ncong$	`\ncong`
$\nshortparallel$	`\nshortparallel`	$\nparallel$	`\nparallel`	$\nvDash$	`\nvDash`
$\nVDash$	`\nVDash`	$\ntriangleright$	`\ntriangleright`	$\ntrianglerighteq$	`\ntrianglerighteq`
$\nsupseteq$	`\nsupseteq`	$\nsupseteqq$	`\nsupseteqq`	$\supsetneq$	`\supsetneq`
$\varsupsetneq$	`\varsupsetneq`	$\supsetneqq$	`\supsetneqq`	$\varsupsetneqq$	`\varsupsetneqq`

5.3.9 Miscellaneous Symbols

There are also some symbols that do not fit into any of the categories above, so we collectively present them in Tables 5.16 and 5.17. It is rather important to stress that the symbol $\mho$ presented in Table 5.16 is the archaic term for the unit of electrical conductance. The modern name of this unit is siemens (symbolized S).

Table 5.23: Continued (part 2).

$\mappedfrom$	`\mappedfrom`	$\longmappedfrom$	`\longmappedfrom`	$\Mapsto$	`\Mapsto`
$\Longmapsto$	`\Longmapsto`	$\Mappedfrom$	`\Mappedfrom`	$\Longmappedfrom$	`\Longmappedfrom`
$\mmapsto$	`\mmapsto`	$\longmmapsto$	`\longmmapsto`	$\mmappedfrom$	`\mmappedfrom`
$\longmmappedfrom$	`\longmmappedfrom`	$\Mmapsto$	`\Mmapsto`	$\Longmmapsto$	`\Longmmapsto`
$\Mmappedfrom$	`\Mmappedfrom`	$\Longmmappedfrom$	`\Longmmappedfrom`	$\varparallel$	`\varparallel`
$\varparallelinv$	`\varparallelinv`	$\nvarparallel$	`\nvarparallel`	$\nvarparallelinv$	`\nvarparallelinv`
$\colonapprox$	`\colonapprox`	$\colonsim$	`\colonsim`	$\Colonapprox$	`\Colonapprox`
$\Colonsim$	`\Colonsim`	$\doteq$	`\doteq`	$\multimapinv$	`\multimapinv`
$\multimapboth$	`\multimapboth`	$\multimapdot$	`\multimapdot`	$\multimapdotinv$	`\multimapdotinv`
$\multimapdotboth$	`\multimapdotboth`	$\multimapdotbothA$	`\multimapdotbothA`	$\multimapdotbothB$	`\multimapdotbothB`
$\VDash$	`\VDash`	$\VvDash$	`\VvDash`	$\cong$	`\cong`
$\preceqq$	`\preceqq`	$\succeqq$	`\succeqq`	$\nprecsim$	`\nprecsim`
$\nsuccsim$	`\nsuccsim`	$\nlesssim$	`\nlesssim`	$\ngtrsim$	`\ngtrsim`
$\nlessapprox$	`\nlessapprox`	$\ngtrapprox$	`\ngtrapprox`	$\npreccurlyeq$	`\npreccurlyeq`
$\nsucccurlyeq$	`\nsucccurlyeq`	$\ngtrless$	`\ngtrless`	$\nlessgtr$	`\nlessgtr`
$\nbumpeq$	`\nbumpeq`	$\nBumpeq$	`\nBumpeq`	$\nbacksim$	`\nbacksim`
$\nbacksimeq$	`\nbacksimeq`	$\neq$	`\neq`, `\ne`	$\nasymp$	`\nasymp`
$\nequiv$	`\nequiv`	$\nsim$	`\nsim`	$\napprox$	`\napprox`
$\nsubset$	`\nsubset`	$\nsupset$	`\nsupset`	$\nll$	`\nll`
$\ngg$	`\ngg`	$\nthickapprox$	`\nthickapprox`	$\napproxeq$	`\napproxeq`
$\nprecapprox$	`\nprecapprox`	$\nsuccapprox$	`\nsuccapprox`	$\npreceqq$	`\npreceqq`
$\nsucceqq$	`\nsucceqq`	$\nsimeq$	`\nsimeq`	$\notin$	`\notin`
$\notni$	`\notni`, `\notowns`	$\nSubset$	`\nSubset`	$\nSupset$	`\nSupset`
$\nsqsubseteq$	`\nsqsubseteq`	$\nsqsupseteq$	`\nsqsupseteq`	$\coloneqq$	`\coloneqq`
$\eqqcolon$	`\eqqcolon`	$\coloneq$	`\coloneq`	$\eqcolon$	`\eqcolon`
$\Coloneqq$	`\Coloneqq`	$\Eqqcolon$	`\Eqqcolon`	$\Coloneq$	`\Coloneq`
$\Eqcolon$	`\Eqcolon`	$\strictif$	`\strictif`	$\strictfi$	`\strictfi`
$\strictiff$	`\strictiff`	$\circledless$	`\circledless`	$\circledgtr$	`\circledgtr`
$\lJoin$	`\lJoin`	$\rJoin$	`\rJoin`	$\Join$	`\Join`, `\lrJoin`
$\openJoin$	`\openJoin`	$\lrtimes$	`\lrtimes`	$\opentimes$	`\opentimes`

➤ **Exercise 5.2** Define a new function name for the Greek version of cos (use the "word" `sun` for the name of the function). □

5.4.3 One Above the Other

There are cases where we want to put one or more objects above one another. If there is a "main" object and a secondary one that needs to be put above the main object in order to characterize it, then we use `\stackrel`, which has two arguments:

$$f(x)\stackrel{\mathrm{def}}{=}\cos x$$

```
\begin{displaymath}
f(x)\stackrel{\mathrm{def}}{=}\cos x
\end{displaymath}
```

If we look carefully at the equation above, we will notice that the first argument of the `\stackrel` command is set using a smaller type size and the second argument is set at the baseline (or as it would be put if it was set alone).

There are cases where one wants to typeset objects one above the other, treating them in the same way; here is an example:

$$\sum_{{I\subseteq \{1,2,\ldots,n\} }\atop {|I|\geq k}} a_I$$

```
\begin{displaymath}
\sum_{{I\subseteq \{1,2,\ldots,n\} }
\atop {|I|\geq k}} a_I
\end{displaymath}
```

As is evident, we use `\atop` to achieve the desired effect. This command is a primitive TEX infix operator and must be used in a local scope.

➤ **Exercise 5.3** Write the code that sets the following display:

$$\sum_{\substack{0\leq i\leq r\\ 0\leq j\leq s\\ 0\leq k\leq t}} C(i,j,k)$$

□

Another need that often arises is to get the result of the `\stackrel` command but with the first argument *below* the second argument. For instance, one may want to write

$$a_n \mathrel{\mathop{\longrightarrow}\limits_{n\to\infty}} 0.$$

In order to achieve this effect, we can define a new command:

```
\newcommand{\abottom}[2]{\mathrel{\mathop{#2}\limits_{#1}}}
```

Here is the code that has been used to typeset the display above:

```
\begin{displaymath}
a_n\abottom{n\to\infty}{\longrightarrow} 0
\end{displaymath}
```

Sometimes, there is a need to put one object above the other and to surround the whole construction with special delimiters. We can get this effect by using the primitive TEX command `\atopwithdelims`. The syntax is shown in the next example. Here are a couple of examples that demonstrate the use of this command:

${\alpha \atopwithdelims<> \beta}$	`$\alpha \atopwithdelims<> \beta$`
${\alpha \atopwithdelims\lfloor\rfloor \beta}$	`$\alpha \atopwithdelims\lfloor\rfloor \beta$`

An alternative way to write ${a\choose b}$ is `$a\choose b$`. The command `\choose` is provided by plain TEX and remains valid in LATEX.

Similar commands for setting fractions with delimiters are the `\overwithdelims` command, which has the same syntax as the `\atopwithdelims`, and the `\abovewith-delims` command, which has a third argument that specifies the width of the fraction line. Both are primitive TEX commands. Here is an example:

${a \overwithdelims() b}$	`$a \overwithdelims() b$`
${a \abovewithdelims() 1pt b}$	`$a \abovewithdelims() 1pt b$`

Other stacking operations are those of underlining and overlining as well as those that put another symbol (e.g., a brace) above or below a collection of objects. In order to set such constructs, we have to use the commands `\underline`, `\overline`, and some others that are shown below:

$\overline{\overline{x}^2}$	`$\overline{\overline{x}^2}$`
$\underline{\underline{x}+\underline{y}}$	`$\underline{\underline{x}+\underline{y}}$`
$\widetilde{xyz}$	`$\widetilde{xyz}$`
$\widehat{xyz}$	`$\widehat{xyz}$`
$\overleftarrow{xyz}$	`$\overleftarrow{xyz}$`
$\overrightarrow{xyz}$	`$\overrightarrow{xyz}$`
$\overbrace{a,\ldots,z}$	`$\overbrace{a,\ldots,z}$`
$\underbrace{\alpha,\ldots,\omega}$	`$\underbrace{\alpha,\ldots,\omega}$`

In particular, for the commands `\underbrace` and `\overbrace`, we should add that they can have an index, as the following example shows:

$\overbrace{a,\ldots,z}^{26}$	`$\overbrace{a,\ldots,z}^{26}$`
$\underbrace{\alpha,\ldots,\omega}_{24}$	`$\underbrace{\alpha,\ldots,\omega}_{24}$`

One should avoid using the over- or under-bracing except in display mode, as this construction takes a lot of vertical space. Also, one should notice the difference between $\overline{x}_n$ and $\overline{x_n}$ (which were produced with `$\overline{x}_n$` and `$\overline{x_n}$`, respectively). Finally, notice that there is often a problem with the "overarrows." They will usually touch (or even cross) the letter z in the example above if special care is not taken. This problem is even worse with capital letters. It is a font design issue and is corrected for the default fonts if the package lamsarrow by Michael Spivak is used (as we did here).

5.4.4 Horizontal Space

When we write mathematical text, LaTeX makes decisions about the spacing of the several mathematical symbols. However, there are cases where one wants to change the default behavior. The following display shows the common commands for changing the spacing in math manually.

$x \qquad x$	`$x \qquad x$`
$x \quad x$	`$x \quad x$`
$x \; x$	`$x \; x$`
$x \: x$	`$x \: x$`
$x \, x$	`$x \, x$`
$x x$	`$x x$`
$x \! x$	`$x \! x$`
$x \!\! x$	`$x \!\! x$`

The command `\!` stands for a negative length and can be used repeatedly, as the last example demonstrates. Of course, we are allowed to use other LaTeX space-changing commands (see Section 6.3.2).

5.4.5 Integrals and Series

Integrals are produced by the command `\int` and sums by the command `\sum`. The end points of integration or summation are declared by providing indices and exponents to these commands. Here are two examples:

$\int_0^\infty e^{-x^2}\,dx=\sqrt{2\pi}$	`$\int_0^\infty e^{-x^2}\,dx=\sqrt{2\pi}$`
$\sum_{n=1}^\infty \frac1{n^2} =\frac{\pi^2}6$	`$\sum_{n=1}^\infty \frac1{n^2} =\frac{\pi^2}6$`

The position of the range of the sum index is not always as we described above. These are put above and below the sum only when we are in display mode. When we are in the first math mode, these are put as indices and exponents like this: $\sum_{n=1}^\infty 1/n^2 = \pi^2/6$. The purpose of this is to avoid forcing the spreading of the text in order to accommodate the summation indices and should not be bypassed. However, if we insist, we may do it using the `\limits` command. Once again, let us stress that this is poor practice, as it affects, in a bad way, the color balance of the page. Take a look at the following example:

However, there are cases where one wants to use another symbol that is not of variable size and pretend it is. For example, one may want to use a symbol such as

$$\mathop{\LARGE\ast}\limits_{i=1}^n f_i$$

for the convolution of functions. For such a task, we need to define a new operator (such as the `\sum` operator), but since the asterisk that we used above is not of variable size, we must decide what size we want. Therefore, the example above was typeset using

```
\newcommand{\astop}{\mathop{\LARGE\mathrm{\ast}}}
.....................................
$$\astop\limits_{i=1}^n f_i.$$
```

5.4.6 Matrices, Arrays, and Nonanalytically Defined Functions

The way to write a matrix in LaTeX is to use the `array` environment. This environment arranges the elements of the matrix in rows and columns without taking care of the delimiters. If we want to use, say, the symbols [ ] for the matrix delimiters, LaTeX must be informed that the size of these delimiters must be adjusted to fit nicely with the matrix they enclose. This is done by the `\left` and `\right` commands. These commands take care of the automatic size adjustment of the delimiters. Here is the syntax for a matrix:

$$\left[\begin{array}{lcr} x-\lambda & 1 & 0 \\ 0 & x-\lambda & 1 \\ 0 & 0 & x-\lambda \end{array}\right]$$

```
$$\left[
  \begin{array}{lcr}
     x-\lambda & 1         & 0         \\
     0         & x-\lambda & 1         \\
     0         & 0         & x-\lambda
   \end{array}
  \right]$$
```

There are cases where we want only one of the two delimiters, and this is the case for the nonanalytically defined functions. If we just omit one of the two delimiters, LaTeX will complain about improper grouping. That is why the `\left` and `\right` commands accept the dot as a special delimiter that produces nothing in the output but satisfies LaTeX's fussiness about proper grouping. Here is an example:

$$\chi_A(x) = \left\{\begin{array}{ll} 1 & \mathrm{if}\ x\in A \\ 0 & \mathrm{otherwise} \end{array}\right.$$

```
$$\chi_A (x) = \left\{
       \begin{array}{ll}
           1 & \mathrm{if}\ x\in A \\
           0 & \mathrm{otherwise}
       \end{array}
  \right.$$
```

There are cases where this simple construction is not satisfactory. Here is an example:

$$a_n = \left\{ \begin{array}{ll} 1+2+3+\cdots+n, & \text{if } n \text{ is even} \\ 0, & \text{otherwise.} \end{array} \right.$$

It would look much better if most of the white space of the second line of the array was not there. For instance,

$$a_n = \left\{ \begin{array}{l} 1+2+3+\cdots+n, \quad \text{if } n \text{ is even} \\ 0, \quad \text{otherwise} \end{array} \right.$$

was produced by putting a `\quad` after the comma and placing everything from the comma to the end of the line in a `\mbox`. In addition, the `array` now has only one left-justified column.

Sometimes, we need to write a matrix with rows and columns being labeled with the number of the row or column. This is done with the command `\bordermatrix`. Here is an example:

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & a_{11} & a_{12} & a_{13} \\ 2 & a_{21} & a_{22} & a_{23} \\ 3 & a_{31} & a_{32} & a_{33} \end{array}$$

```
$$\bordermatrix{ & 1 & 2 & 3 \cr
1 & a_{11} & a_{12} & a_{13} \cr
2 & a_{21} & a_{22} & a_{23} \cr
3 & a_{31} & a_{32} & a_{33}}$$
```

The `\bordermatrix` command is not a LaTeX command, but it comes from plain TeX. That is why its syntax is not similar to the way that we created matrices and it uses the `\cr` command to denote the end of the line instead of the double backslash. Plain TeX provides also a `\matrix` command (with similar syntax) for matrices, but its use is obsolete in LaTeX. It can be used, but often it creates problems that cannot be overcome and that is why we do not describe it. Similarly, plain TeX provides the command `\cases` for writing nonanalytically defined functions and many mathematicians use it. Once again, the behavior of this command in LaTeX is unpredictable and very often does not work.

The `array` environment takes an optional argument that can be either `t` or `b`. These options align the top or the bottom of the array block to the baseline instead of centering it, which is the default. For example, the matrices

$$\left(\begin{array}[t]{lcr} x-\lambda & 1 & 0 \\ 0 & x-\lambda & 1 \\ 0 & 0 & x-\lambda \end{array} \right) \left(\begin{array}[b]{lcr} x-\lambda & 1 & 0 \\ 0 & x-\lambda & 1 \\ 0 & 0 & x-\lambda \end{array} \right)$$

were produced by using the `lcr` ***`table specs`*** and the `t` and `b` optional arguments, respectively. Notice that the parentheses are more cursive than the standard ones of

Note that the frame in all examples of this section is not produced by the code presented. We see that the defaults for the theorem environments are as follows. The `output name` is written in bold, then the number follows, which is the section's number, followed by the theorem's counter. The alternate position of the optional argument `[counter]` for the lemmata, corollaries, and so on, says that the counters for these are the same as the theorem's counter, which, in turn, is defined from the section's number *followed* by the theorem counter. If, for example, we had set `\newtheorem{lemma}{Lemma}[thm]`, the number of our lemma above would not be 5.4.2 but 5.4.1.1. In other words, the theorem's number is *followed* by the theorem's counter. After that, the theorem's text is set in italics.

There is one optional argument for all of the theorem environments that is used to typeset the name of the people who proved the theorem or a theorem's name, if such a name exists. For example, we could write

```
\begin{thm}[Euclid]
There are infinitely many primes.
\end{thm}
```

to get

Theorem 5.4.3 (Euclid) *There are infinitely many primes.*

or, if we are in a multilingual environment we can set the language with the relative command. For example, the code

```
\begin{thm}[\textgreek{E>υκλεíδης}]
  There are infinitely many primes.
  \end{thm}
```

can be used to get

Theorem 5.4.4 (Εὐκλείδης) *There are infinitely many primes.*

Sometimes, it happens that immediately after the theorem header, we need a line break. This happens when the theorem's description is long and the first word of its statement does not fit on the line and cannot be hyphenated (it may be one syllable). This problem appears again when we typeset with a smaller text width, as in a multicolumn environment. The way to achieve this line break is to use an `\hfill` command after the `\begin{theorem}` and leave a blank line after that, like this:

Naturally, we can customize the amount of white space that LaTeX puts around the aligned symbol. To do this, we simply set the length variable `\arraycolsep` in a local scope that encloses the `eqnarray` environment, which is what we did in the next example. Of course, if we set this variable in our document's preamble, then the effect is global.

Sometimes, we want to write such an equation array but do not want to number all of the equations. This can be done with the `\nonumber` command:

$$\begin{aligned}\int_0^{\pi/2}\sin x\,dx &= -\cos x\biggm|_0^{\pi/2} && (5.6)\\ &= -\cos\frac{\pi}{2}-(-\cos 0)\\ &= -0-(-1)\\ &= 1\end{aligned}$$

```
\begin{eqnarray}
\int_0^{\pi/2} \sin x \,dx
&=& -\cos x\biggm|_0^{\pi/2}\\
&=& -\cos\frac{\pi}2
    -(-\cos 0) \nonumber \\
&=& -0-(-1) \nonumber \\
 &=& 1 \nonumber
\end{eqnarray}
```

However, we should use the `eqnarray*` environment if we want no numbers at all. One may note that the command `\biggm` above is not yet discussed. This is the subject of the next section.

Now, we will see how to typeset logic proofs since these require more attention. We need the `proof` package (by Makoto Tatsuta). The package provides the command `\infer[`*`label`*`]{`*`lower`*`}{`*`upper`*`}`, which draws an inference labeled with *label*. If we put an asterisk after the command (`\infer*[...]`), it draws a many-step deduction. If the star is changed to the = sign, it draws a double-ruled deduction. It also provides the command `\deduce[`*`proof`*`]{`*`lower`*`}{`*`upper`*`}`, which draws an inference without a rule with a *proof* name.

Thus, `\infer{A}{B}` and `\deduce{A}{B}` produce $\frac{B}{A}$ and $\genfrac{}{}{0pt}{}{B}{A}$. To produce many steps, we just use the alignment character `&`, and the code for the first equation of the display

$$\pi = \cfrac{D \quad \genfrac{}{}{}{}{\overline{F \,\&\, G \,\&\, H}}{E}\,(\to I)}{A \,\&\, B \,\&\, C}\,(\& I) \qquad\qquad \cfrac{\genfrac{}{}{0pt}{}{C \,\&\, D}{\vdots} \\ B \qquad E}{A} \qquad (5.7)$$

is

```
\[  \pi = \vcenter{\infer[(\& I)]{A\,\&\,B\,\&\,C}{D &
             \infer=[(\to I)]{E}{F\,\&\,G\,\&\,H}\hspace{2em}}}\]
```

where `\vcenter` centers its argument vertically to the baseline and `\hspace` is used in order to make the second top argument bigger to force a bigger inference line. The code for the second equation is standard LaTeX `$\hat{\hat A}$`: $\hat{\hat{A}}$, $\mathcal{AMS}$ `$\hat{\hat A}$`: $\hat{\hat{A}}$.

The same holds true for all other accents (see Table 5.3). Double accents take a lot of processing time, and this is why, if we use them repeatedly, it is better to store the result of a double accent to a command using the `\accentedsymbol` available with the amsxtra package. This command introduces a shorthand and should be used only in the document's preamble. Here is an example:

$\hat{\hat{A}}$ and $\dot{\breve{Y}}$

```
\accentedsymbol{\Ahathat}{%
\hat{\hat{A}}}
\accentedsymbol{\Ybrevedot}{%
\dot{\breve{Y}}}
............................
$\Ahathat$ and $\Ybrevedot$
```

The commands `\dddot` and `\ddddot` produce triple and quadruple dot accents in addition to the `\dot` and `\ddot` accents (which are already available with standard LaTeX): `$\dddot{E}$` and `$\ddddot{T}$` give $\dddot{E}$ and $\ddddot{T}$, respectively.

Special symbols that are set as superscripts form another kind of accent. These are useful in math (for instance the Fourier transform uses a `\hat` as superscript unless the function is a single letter or a few letters). For example,

```
$\bigl(\exp(-x^2)\bigr)\sphat$
```

gives $\bigl(\exp(-x^2)\bigr)\widehat{}$. Notice that we do not use the ^ character. The reader is recommended to try the commands `\spcheck`, `\sptilde`, `\spdot`, `\spddot`, `\spdddot`, and `\spbreve`. All of them are available with the amsxtra package.

5.5.3 Dots

The $\mathcal{AMS}$ packages provide five commands for accessing differently positioned ellipsis dots. `\dotsc` represents "dots with commas" like this $1,2,\dots,n$ (`$1,2,\dotsc,n$`). `\dotsb` stands for "dots with binary operators/relations" as in $1+2+\cdots+n$ (`$1+2+ \dotsb +n$`). `\dotsm` stands for "multiplication dots" as in $a_1a_2\cdots a_n$ (`$a_1 a_2 \dotsm a_n$`). `\dotsi` stands for "dots with integrals" as in $\int_A\int_B\cdots$ (`$\int_A\int_B \dotsi$`). Finally, `\dotso` covers "other dots," which are none of the above: $a\dots b+\dots+c$ (`$a\dotso b+\dotso +c$`).

5.5.4 Nonbreaking Dashes

There are cases (such as when we give the page range of a reference) when we do not want to allow a line break at the en dash point. This can be done with the command `\nobreakdash`. So, if you write "pages 321–345" as `pages 321\nobreakdash--345`, a line break will never occur between the dash and 345. The command can also be used for combinations such as `$p$-adic`. Naturally, one can define shorthands for commonly used constructs, but this is the subject of the next chapter.

5.5.8 Extensible Arrows

The commands `\xleftarrow` and `\xrightarrow` provide extensible arrows in order to accommodate expressions above and below them:

$0\xleftarrow{x\to -\infty} f(x) \xrightarrow[x\to\infty]{x\notin\mathbb{Q}} 1$

```
$0\xleftarrow{x\to -\infty} f(x)
\xrightarrow[x\to\infty]{x\notin\mathbb{Q}} 1$
```

5.5.9 Affixing Symbols to Other Symbols

Standard LaTeX provides the `\stackrel` command for placing something above a binary relation. The $\mathcal{AMS}$ packages provide more general commands, `\overset` and `\underset`. These work with anything and not only with binary relations:

$\overset{\circ}{\textrm{X}}$
$\underset{\ast}{\textrm{X}}$

```
$\overset{\circ}{\textrm{X}}$
$\underset{\ast}{\textrm{X}}$
```

5.5.10 Fractions and Related Constructs

The command `\genfrac` provides an easy interface to define new fractions. Its syntax is as follows:

```
\genfrac{left-delim}{right-delim}{line-thickness}
        {dtyle}{numerator}{denominator}
```

The left and right delimiters are used, for example, for binomial expressions. The line thickness refers to the fraction line and is set to 0 pt for binomial expressions. To select the style, we use a number from 0 to 3. The number 0 is for display style, 1 for text style, 2 for script style, and 3 for script-script style.

By default, the following commands are defined:

Command	Expansion
`\tfrac{x}{y}`	`\genfrac{}{}{}{1}{x}{y}`
`\dfrac{x}{y}`	`\genfrac{}{}{}{0}{x}{y}`
`\binom{x}{y}`	`\genfrac{(}{)}{0pt}{}{x}{y}`
`\dbinom{x}{y}`	`\genfrac{(}{)}{0pt}{0}{x}{y}`
`\tbinom{x}{y}`	`\genfrac{(}{)}{0pt}{1}{x}{y}`

The commands `\tfrac` and `\dfrac` provide convenient abbreviations for `{\textstyle\frac{..}{..}}` and `{\displaystyle\frac{..}{..}}`, respectively. Here is an example:

$$\tfrac{1}{x}\log x$$

```
$$\tfrac{1}{x}\log x$$
```

$$\dfrac{1}{x}\log x$$

```
$$\dfrac{1}{x}\log x$$
```

Here is an example of `\dbinom` and `\tbinom`:

$$\dbinom{n}{k}+\frac{\tbinom{n}{k}}{k!}$$

```
$$\dbinom{n}{k}+
\frac{\tbinom{n}{k}}{k!}$$
```

The special command `\cfrac` is for writing continued fractions:

$$\cfrac{1}{a+\cfrac{1}{a+\cfrac{1}{a+{\ddots}}}}$$

```
$$
\cfrac{1}{a+
  \cfrac{1}{a+
    \cfrac{1}{a+{
      \above0pt \ddots}
}}}
$$
```

You can request that the numerators to be set to the left or right of the fraction line. This is accomplished by using `\cfrac[l]` or `\cfrac[r]`.

5.5.11 The `\smash` Command

The `\smash` command zeros the depth (option `b`) or height (option `t`) of characters and is useful for alignments. In the following example we present two different formulas typeset using the `\smash` command (odd rows) and without using the `\smash` command (even rows). The reader should have a close look at the result to see the difference.

$\sqrt{x}+\sqrt{\smash[b]{y}}+\sqrt{z}$

$\sqrt{x}+\sqrt{y}+\sqrt{z}$

$(1-\sqrt{\smash[b]{\lambda_j}})X$

$(1-\sqrt{\lambda_j})X$

```
$\sqrt{x}+\sqrt{\smash[b]{y}}+\sqrt{z}$
$\sqrt{x}+\sqrt{y}+\sqrt{z}$
$(1-\sqrt{\smash[b]{\lambda_j}})X$
$(1-\sqrt{\lambda_j})X$
```

5.5.12 Operator Names

We saw in Section 5.4.2 how to define new functions/operators with standard LaTeX. The $\mathcal{AMS}$ packages provide an easy interface for this. If you want to define the operator `\random`, all you have to say is

```
\DeclareMathOperator{\random}{random}
```

There is also a starred form:

```
\DeclareMathOperator*{\Lim}{Lim}
```

This means that the defined operator should have subscripts and superscripts placed in the "limits" positions (above and below like, say, the `\max` operator).

In addition to the ones already predefined by standard LaTeX (see Table 5.24), we also have the following available:

`\injlim` $(\operatorname{inj\,lim})$	`\lg` $(\lg)$	`\projlim` $(\operatorname{proj\,lim})$	`\varlimsup` $(\varlimsup)$
`\varliminf` $(\varliminf)$	`\varinjlim` $(\varinjlim)$	`\varprojlim` $(\varprojlim)$	

There is also the command `\operatorname` such that `\operatorname{xyz}` can be used as a binary operator. You can use `\operatorname*` in order to get limits.

5.5.13 The `\mod` Command and its Relatives

The several space conventions for the mod notation are handled by the commands `\mod`, `\bmod`, `\pmod`, and `\pod`. The second and third commands are available in standard LaTeX as well. Here is an example:

$\gcd(m,n\bmod n)$	`$\gcd(m,n\bmod n)$`
$x\equiv y \pmod b$	`$x\equiv y \pmod b$`
$x\equiv y\mod c$	`$x\equiv y\mod c$`
$x\equiv y \pod d$	`$x\equiv y \pod d$`

5.5.14 The `\text` Command

The command `\text` is provided for writing text in math mode. If the text is to be written in sub/super-script position, the text size is adjusted automatically, and this is its main advantage over the previously described method using a `\mbox`:

$f(x)\stackrel{\text{def}}{=} x^2$	`$$f(x)\stackrel{\text{def}}{=} x^2$$`

In a multilingual environment, the command will use the current text language and will accept language-specific commands.

5.5.15 Integrals and Sums

We have seen how to deal with stacked expressions under a `\sum` symbol using the `\atop` command. The $\mathcal{A}\kern-.1667em\lower.5ex\hbox{$\mathcal{M}$}\kern-.125em\mathcal{S}$ packages provide the command `\substack` and the slightly more general environment `subarray`, which has a column specifier:

$$\sum_{\substack{n\in\mathbb Z\\ n\geq 0}} f(n)$$

```
\begin{displaymath}
\sum_{\substack{
 n\in\mathbb Z\\
 n\geq 0}} f(n)
\end{displaymath}
```

$$\sum_{\begin{subarray}{l} n\in\mathbb Z\\ 0\leq n\leq k! \end{subarray}} f(n)$$

```
\begin{displaymath}
\sum_{%
\begin{subarray}{l}
 n\in\mathbb Z\\
 -k!\leq n\leq k!
 \end{subarray}} f(n)
\end{displaymath}
```

If one wants to put accents and limits on a large operator, he or she can use the command `\sideset`. Here is an example that fully demonstrates the capabilities of this command:

$$\sideset{_1^2}{_3^4}\sum_a^b$$

```
$$\sideset{_1^2}{_3^4}\sum_a^b$$
```

5.5.16 Commutative Diagrams

Commutative diagrams are supported with the amscd package. This is provided not as a complete solution but as a package for a quick diagram, drawn without diagonal arrows (for a complete solution, see Section 5.4.11). Consequently, we will not go to the trouble to describe the functionality of this package. Here is an example that demonstrates the use of the package:

$$\begin{tikzcd}
A \arrow[r, "a", "b"'] \arrow[d, "c"'] & B \\
C \arrow[r, equal] & D \arrow[u, "d"']
\end{tikzcd}$$

```
\beg{displaymath}
 \begin{CD}
   A     @>a>b> B   \\
   @VcVV       @AAdA\\
   C     @=    D
 \end{CD}
\end{displaymath}
```

5.5.17 Displayed Equations and Aligned Structures

Maybe the biggest advantage of using the $\mathcal{AMS}$ packages is the ability they give to better deal with displayed and aligned environments—much better than the already discussed eqnarray environment of LaTeX. These environments are:

equation	equation*	align	align*
gather	gather*	flalign	flalign*
multline	multline*	alignat	alignat*
split	gathered	aligned	alignedat

The *space-above* is the space between the theorem and the last line of the previous paragraph. The *space-below* is the space between the theorem and the next paragraph. If you leave these two empty, then the "usual" space will be used. The *body-font* declaration needs no explanation (you can use, for example, `\itshape`). The *indent amount* is the indentation space before the header begins. If you leave this empty, then no indentation will be used. You can also use an already defined length such as `\parindent` for paragraph-like indentation. For example, *Thm head font* can be set to `\bfseries` and the *punctuation after Thm head* is the punctuation that will be set after the theorem (for example, a dot: {.}). Finally, *space after Thm head* is self-explanatory (if set to `\newline` it will create a line break after the theorem head) and the *Thm head spec*, if set to `[\thmnot#3]`, will produce the comment in the theorem's header, read from the square brackets of `\begin{thm}[Thm Head spec]`.

The `proof` environment has an optional argument that customizes the head of the proof. For example, you may say

```
\begin{proof}[Proof of the main theorem]
```

The shape of the QED symbol is controlled by `\qedsymbol`. The default is □. It can also be accessed, if necessary, by the `\qed` command. It frequently happens in mathematics that a proof may end with a display equation or array. In these cases, the position of the QED symbol is problematic. The nice way is to use the command `\qedhere` at the end of the line where the display ends. For example,

Proof.

$$\begin{aligned} f(x) &= x - x \\ &= 0 \qquad \square \end{aligned}$$

was produced by

```
\begin{proof}
  \begin{align*}
      f(x) &=x-x\\
             &=0\qedhere
  \end{align*}
\end{proof}
```

It is important to note that the `amsthm` package must be loaded *before* the `amsmath` package. It is automatically loaded with the $\mathcal{AMS}$ document classes. In case you get an error relating to the `\qedhere` command, try `\mbox{\qedhere}` instead.

5.5.22 Options of the **amsmath** Package

The following options are available for the amsmath style file:

(time/temperature pairs) that we will use are: $(1,65)$, $(2,69)$, $(3,74)$, $(4,79)$, $(5,83)$, $(6,86)$, $(7,88)$, $(8,90)$, $(9,90)$, $(10,90)$, $(11,89)$, $(12,88)$, $(13,85)$, $(14,80)$, $(15,76)$, $(16,74)$, $(17,73)$, $(18,72)$, $(19,71)$, $(20,69)$, $(21,68)$, $(22,67)$, $(23,67)$, and $(24,66)$. Moreover, we use the formulas $x = t \cdot 1000/25$ and $y = (T - 65) \cdot 1000/(95 - 65)$ to scale our data for plotting.

This time, we begin by defining a plotting symbol to save typing later, and to allow for easy subsequent changes in our plotting symbol just by changing a single statement.

```
\newcommand{\plotsymbol}{%
  \put(0,0){\small \makebox(0,0){+}}}
```

Then, the basic picture environment is defined with a unit length of 0.08 mm

```
{\sffamily
\setlength{\unitlength}{0.08mm}
\begin{picture}(1100,1100)
     .............
\end{picture}}
```

and we proceed by inserting picture-drawing statements in the body of the `picture` environment, as before. First, we add some axes

```
\put(0,0){\line(1,0){1000}}
\put(0,0){\line(0,1){1000}}
```

followed by tick marks on the axes

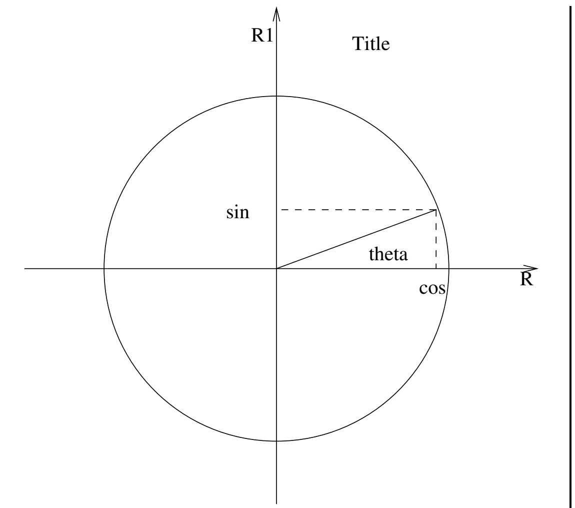
```
\multiput(0,-10)(200,0){6}{\line(0,1){10}}
\multiput(-10,0)(0,189){6}{\line(1,0){10}}
```

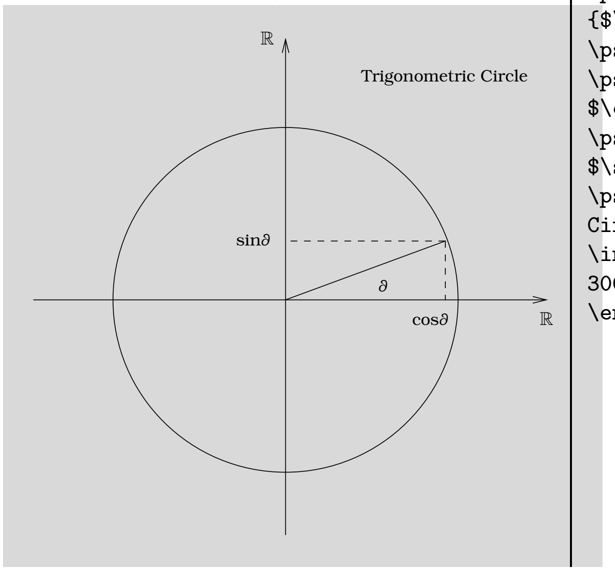
We insert the commands for the x-axis numerals, each numeral is centered over its coordinate by enclosing it in a `\makebox` of zero width

```
\put(0,-40){\makebox(0,0){{\footnotesize 0}}}
\put(200,-40){\makebox(0,0){{\footnotesize 5}}}
\put(400,-40){\makebox(0,0){{\footnotesize 10}}}
\put(600,-40){\makebox(0,0){{\footnotesize 15}}}
\put(800,-40){\makebox(0,0){{\footnotesize 20}}}
\put(1000,-40){\makebox(0,0){{\footnotesize 25}}}
```

and the commands for the y-axis numerals

```
\put(-40,0){\makebox(0,0){{\footnotesize 65}}}
\put(-40,189){\makebox(0,0){{\footnotesize 70}}}
\put(-40,379){\makebox(0,0){{\footnotesize 75}}}
\put(-40,568){\makebox(0,0){{\footnotesize 80}}}
\put(-40,757){\makebox(0,0){{\footnotesize 85}}}
\put(-40,947){\makebox(0,0){{\footnotesize 90}}}
```





```
\documentclass{article}
\usepackage{kerkis,kmath}
\pagestyle{empty}
\usepackage{amsfonts,graphicx}
\usepackage{psfrag}
\begin{document}
\psfrag{R}{\hspace{.6em}\raisebox%
{-1ex}{$\mathbb R$}}
\psfrag{R1}{\raisebox{3.5ex}%
{$\mathbb R$}}
\psfrag{theta}{$\theta$}
\psfrag{cos}{\hspace{-1em}\mbox{%
$\cos\theta$}}
\psfrag{sin}{\raisebox{.5ex}{%
$\sin\theta$}}
\psfrag{Title}{Trigonometric
Circle}
\includegraphics[bb=80 -10 %
300 280]{trigcircle.eps}
\end{document}
```

Figure 9.5: The original drawing (top), the modified output (bottom), and the LaTeX code that modified the original drawing. Here, we use the experimental packages kerkis and kmath, which support the Kerkis typeface. However, one gets similar results when using either the standard typeface or any other typeface.

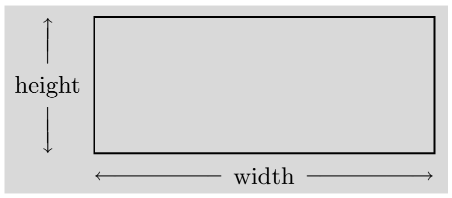
```
\setplotsymbol ({plot symbol}[o_x o_y] <xshift,yshift>)
```

The parameters surrounded by square brackets and by < > are optional and have the expected meaning.

If we want to place some text between arrows (i.e., to label something), we can use the `\betweenarrows` command

```
\betweenarrows {text}[o_x o_y] <xshift,yshift>
  from xcoord_s ycoord_s xcoord_e ycoord_e
```

As above, the parameters surrounded by square brackets and by < > are optional. The following is a simple example:

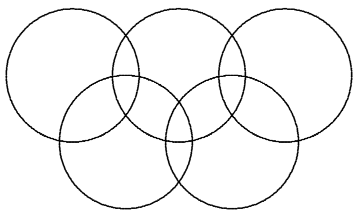


```
\setcoordinatesystem
units <3cm,3cm>
\putrectangle corners
at 0 0 and 1.5 .6 \small
\betweenarrows {width} [t]
<0pt,-5pt> from  0 0 to 1.5 0
\betweenarrows {height} [r] <
-5pt,0pt> from 0 0 to 0 .6
```

PiCTeX provides commands that can be used to draw arcs of circles or ellipses. The command

```
\circulararc θ degrees from xcoord_s ycoord_s
  center at xcoord_c ycoord_c
```

draws an arc of a circle centered at point $(xcoord_c, ycoord_c)$; the arc starts from $(xcoord_s, ycoord_s)$ and goes counterclockwise by θ degrees. Here is a simple example:



```
\setcoordinatesystem
        units <5pt,5pt>
\multiput {
  \circulararc 360
          degrees from 5 0
    center at 0 0 }
at 0 0 8 0 16 0 4 -5 12 -5 /
```

Similarly, the command

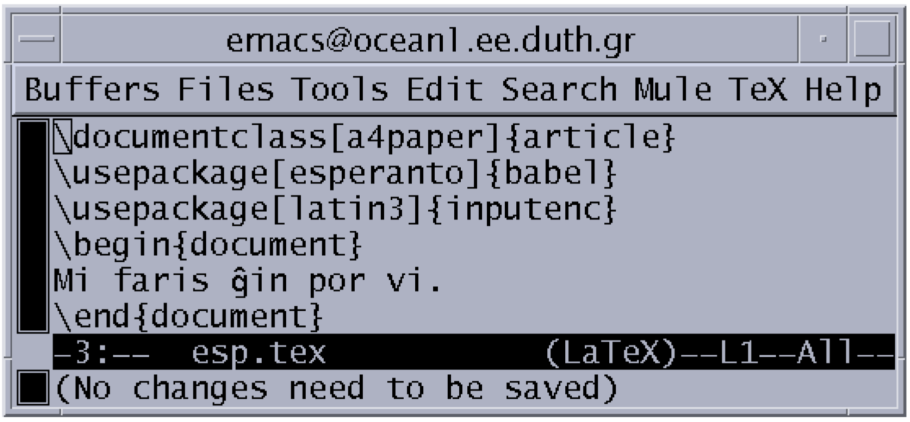
```
\ellipticalarc axes ratio ξ:η θ degrees from xcoord_s ycoord_s
  center at xcoord_c ycoord_c
```

draws an arc of an ellipse whose minor and major axes are parallel to the x and y axes. The numbers ξ and η are proportional to the lengths of the horizontal and vertical axes of the ellipse.

in South Africa. This dialect is also supported by babel (option `afrikaans`). If we do not want to prepare our input file using all of these shorthands, then we simply prepare our input file with an editor that is aware of the ISO-8859-1 character set and use the `t1enc` package. As far as it regards Λ, one should type a document using a Unicode editor to avoid all of these commands. Otherwise, the appropriate ΩCP must be loaded. And since Ωbabel is still far away, one has to redefine the commands that produce the predefined name strings such as "chapter," "preface," and so on.

10.7 The Esperanto Language

Esperanto is an artificial language intended for use between people who speak different native languages. Esperanto was developed during the period 1877–1885 by L.L. Zamenhof of Warsaw, Poland. The `esperanto` option defines all of the language-specific macros for the Esperanto language. The language uses the Latin alphabet and the letters Ĉ, ĉ, Ĝ, ĝ, Ĥ, ĥ, Ĵ, ĵ, Ŝ, ŝ, Ŭ, and ŭ. All of these extra letters can be accessed with the command `"`ℓ, where ℓ is any of the extra letters. For example to typeset the sentence "Mi faris ĝin por vi," we simply type `Mi faris "gin por vi`. Of course, if we type our document with the aid of an editor that supports the Latin 3 encoding, then our input file will contain the extra characters of this encoding. For example, in the following screen dump, it is clear that we type Latin 3 characters and use the necessary input encoding:



```
\documentclass[a4paper]{article}
\usepackage[esperanto]{babel}
\usepackage[latin3]{inputenc}
\begin{document}
Mi faris ĝin por vi.
\end{document}
-3:--  esp.tex          (LaTeX)--L1--All--
(No changes need to be saved)
```

This is in general the mechanism by which we prepare multilingual LaTeX input files. If we want to typeset Esperanto language documents with Λ, we must use the `in88593` ΩCP. Similar ΩTPs exist for other Latin encodings: `in88591` (for Latin 1), `in88592` (for Latin 2), and `in88594` (for Latin 4).

In addition to the coptic package, one can also use the copte package (by Serge Rosmorduc) to typeset Coptic text.

It should be noted that as far as the hieroglyphs are concerned, the package provides access to a small subset of the glyphs of the Sesh Nesout system created by Serge Rosmorduc. Wilson chose about 70 of the most common glyphs from this package to create the hieroglyph package.

Table 10.11 shows the new font selection commands provided by the packages described in this section. The short passage commands take as argument a piece of text

Table 10.11: New commands provided by the packages that provide support for archaic writing systems.

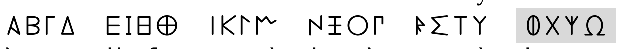
Package	Font Family Selection	Short Passage Command
copte	—	`\textcopte`
cypriot	`\cyprfamily`	`\textcypr`
etruscan	`\etrfamily`	`\textetr`
greek4cbc	`\givbcfamily`	`\textgivbc`
greek6cbc	`\gvibcfamily`	`\textgvibc`
hieroglf	`\pmhgfamily`	`\textpmhg`
linearb	`\linbfamily`	`\textlinb`
oldprsn	`\copsnfamily`	`\textcopsn`
phoenician	`\phncfamily`	`\textphnc`
protosem	`\protofamily`	`\textproto`
runic	`\futfamily`	`\textfut`
ugarite	`\cugarfamily`	`\textcugar`

in the corresponding language. In addition, to typeset Coptic language texts, we need to load the font encodings COP and T1. In other words, we must include the following command in the preamble of our file:

```
\usepackage[COP,T1]{fontenc}
```

In what follows, we present the transliterations defined by the various packages that can be used to access individual letters or symbols, in general.

6th century Greek



```
\textgvibc{ABGD EZH\TTheta\ IKLM N\TXi OP RSTU
           \TPhi X\TPsi\TOmega}
```

4th century Greek (smooth)

ΑΒΓΔ ΕΖΗΘ ΙΚΛΜ ΝΞΟΠ ΡΣΤΥ ΦΧΨΩ

```
\textgivbc{ABGD EZH\TTheta\ IKLM N\TXi OP RSTU
          \TPhi X\TPsi\TOmega}
```

4th century Greek (rough)

ΑΒΓΔ ΕΖΗΘ ΙΚΛΜ ΝΞΟΠ ΡΣΤΥ ΦΧΨΩ

```
\textgivbc{abgd ezh\tTheta\ iklm n\tXi op rstu
          \tPhi x\tPsi\tOmega}
```

Etruscan

𐌀𐌁𐌂𐌃 𐌄𐌅𐌆𐌇 𐌈𐌉𐌊𐌋 𐌌𐌍𐌎𐌏 𐌐𐌑𐌒𐌓 𐌔𐌕𐌖𐌗 𐌘𐌙𐌚 𐌀𐌁𐌂𐌃 𐌄𐌅𐌆𐌇
𐌈𐌉𐌊𐌋 𐌌𐌍𐌎𐌏 𐌐𐌑𐌒𐌓 𐌔𐌕𐌖𐌗 𐌘𐌙𐌚

```
\textetr{ABGD EFZH \TTheta IKL MN\TXi OP\Tsade Q RSTU X\TPhi\TPsi
8 abgd efzh \tTheta ikl mn\tXi op\tsade q rstu x\tPhi\tPsi 8}
```

Phoenician

𐤀𐤁𐤂𐤃 𐤄𐤅𐤆𐤇 𐤈𐤉𐤊𐤋 𐤌𐤍𐤎𐤏 𐤐𐤑𐤒𐤓 𐤔𐤕 𐤀𐤁𐤂𐤃 𐤄𐤅𐤆𐤇 𐤈𐤉𐤊𐤋
𐤌𐤍𐤎𐤏 𐤐𐤑𐤒𐤓 𐤔𐤕

```
\textphnc{ABGD EFZH \TTheta IKL MN\TXi O P\Tsade QR ST
 abgd efzh \tTheta ikl mn\tXi o p\tsade qr st}
```

Runic

ᚠᚢᚦᚨ ᚱᚲᚷᚹ ᚺᚾᛁᛃ ᛇᛈᛉᛊ ᛏᛒᛖᛗ ᛚᛜᛞᛟ :

```
\textfut{FU\Fthorn A RKGW HNIJ YPXS TBEM L\Fng DO :}
```

Ugaritic

𐎀𐎁𐎂𐎃 𐎄𐎅𐎆𐎇 𐎈𐎉𐎊𐎋 𐎌𐎍𐎎𐎏 𐎐𐎑𐎒𐎓 𐎔𐎕𐎖𐎗 𐎘𐎙𐎚𐎛
𐎜𐎝𐎟

```
\textcugar{abgH dhwz IJyk SlmD nZs‘ pxqr TGti uX:}
```

abgh dhwz ḥṭyk lmd nẓsʻ pṣqr tgti us:

```
\translitcugar{\Ua\Ub\Ug\Uhu \Ud\Uh\Uw\Uz  \Uhd\Utd\Uy\Uk
\Usa\Ul\Um\Udb \Un\Uzd\Us\Ulq \Up\Usd\Uq\Ur \Utb\Ugd\Ut\Ui
\Uu\Usg\Uwd}
```

```
\newcommand{\newaddcontentsline}[3]{%
  \addtocontents{#1}{%
 \protect\contentsline{#2}{*#3}{\thepage}}}
\let\oldaddcontentsline\addcontentsline
\newcommand\firstarg{}
\newcommand{\starredsection}[2][]{%
   \let\addcontentsline\newaddcontentsline%
   \renewcommand\firstarg{#1}%
           \ifx \firstarg\empty
      \relax\section{#2}%
   \else%
      \section[#1]{#2}%
   \fi%
   \let\addcontentsline\oldaddcontentsline}
```

Note that we are not allowed to use a local scope in the definition of the new sectioning command, so we have to play with the `\let` command.

2.7 (page 27):

If we put the `abstract` environment before the `\maketitle` command, then the abstract appears on a separate page. On the other hand, if we put the environment after the `\maketitle` command, the abstract appears just after the title of the document.

2.8 (page 28):

We only show the postal code part:

```
671\␣00\␣\␣Xanthi
```

3.1 (page 45):

Note that in order to correctly typeset this text, one has to use both forms of the font selection commands:

```
Because of \texttt{typewriters} and the usually
\textbf{incomplete} fonts that come with
\textsc{word processors}, many people have learned
to \textsl{emphasize} by \textit{underlining}
the text. That is \textbf{really \textit{poor}} as
it disturbs the \textsl{balance} of the
``{\fontshape{ui}\selectfont page color}'' and
makes it look {\sffamily unprofessional}.
```

3.2 (page 45):

Here we used the `\itshape` command to denote natural numbers. We avoided math mode since it is not discussed up to this point:

```
\textbf{Theorem 1} Every natural number bigger than 1 has a
prime divisor.
```