Chapter 2
Running Text

This chapter explains everything you've always wanted to know about writing text, aligning it, and changing text appearance.

Recall from Chapter 1 that LaTeX is implemented on top of TeX, which is a rewriting machine that turns token streams into token streams. Some of the character tokens in the input stream have a special meaning to TeX. This is studied in Section 2.1. The rest of the chapter is about typesetting. We start with some sections about diacritics, ligatures, dashes, emphasis, footnotes and marginal notes, quotes and quotations. If you're not familiar with these notions then don't worry, because they are explained further on. Also you can visit the typography jargon reference on page 267. This chapter ends with sections about changing the size and the type style of the text, the most important text alignment techniques, and language related issues.

2.1 Special Characters

This section studies ten characters that have a special meaning to TeX. When TeX sees these characters as tokens in the input stream, then it usually does not typeset them but, instead, changes state. The remainder of this section briefly explains the purpose of the tokens and how you typeset them as characters in the output.

Table 2.1 depicts the tokens, their meaning, and the command to typeset them. We have already studied the start-of-comment token (%) and the backslash (\), which starts control sequences. Typesetting a backslash is done with the commands `\textbackslash` and `\backslash`. The latter command is only used when specifying mathematical formulae, which is described in Chapter 8. The parameter reference token is described in Chapter 11. The alignment tab (&) is described in Section 2.19.3. This token usually indicates a horizontal alignment position in array-like structures consisting of rows and columns. The math mode switch token ($), the subscript token (_), and the explained token (^) are described in Chapter 8. The three remaining tokens are described in the remainder of this section.

2.1.1 *Tieing Text*

Remember that LaTeX is a large rewriting machine that repeatedly turns token sequences into token sequences. At some stage it turns a

M.R.C. van Dongen, *LaTeX and Friends*, X.media.publishing,
DOI 10.1007/978-3-642-23816-1_2, © Springer-Verlag Berlin Heidelberg 2012

Table 2.1
The characters in the first column have a special meaning to LaTeX. The purpose of the characters is listed in the column 'Purpose.' The last column lists the command that produces the character. The command `\textbackslash` is used when typesetting normal text. The command `\backslash` is used when typesetting mathematics.

Token	Purpose	Command
#	parameter reference	`\#`
$	math mode switch	`\$`
%	start of comment	`\%`
&	alignment tab	`\&`
~	text tie token	`\textasciitilde`
_	math subscript	`\_`
^	math superscript	`\textasciicircum`
{	start of group	`\{`
}	end of group	`\}`
\	start of command	`\textbackslash` or `\backslash`

token sequence into lines. This is where LaTeX (TeX really) determines the line breaks. The tilde token (~) defines an inter-word space that cannot be turned into a line break. As such it may be viewed as an operator that ties words.

The following shows two important applications of the tilde operator: it prevents unpleasant linebreaks in references and citations.

LaTeX Usage
```
... Figure~\ref{fig:list@format}
  depicts the format of a list.
 It is a reproduction of~\cite[Figure~6.3]{Lamport:94}.
```

It is usually not too difficult to decide where to use the tie operator. The following are some concrete examples, which are taken from [Knuth 1990, Chapter 14].

- References to named parts of a document:
  - `Chapter~12`,
  - `Theorem~1.5`,
  - ....

  Knuth [1990] recommends that you use `Lemmas 5 and~6` because having the 5 at the start of a line is not really a problem.
- Between a person's forenames and between multiple surnames:
  - `Donald~E. Knuth`,
  - `Luis~I. Trabb~Pardo`,
  - `Bartel~Leendert van~der~Waarden`,
  - `Charles~XII`,
  - ....
- Between math symbols in apposition with nouns:
  - `dimension~$d$`,
  - `string~$s$ of length~$l$`,
  - ....

  Here the construct `$`⟨math⟩`$` is used to typeset ⟨math⟩ as an in-line mathematical expression.
- Between symbols in series:
  - `1,~2, or~3`.
- When a symbol is a tightly bound object of a preposition:

    - from 0 to~1,
    - increase z by~1,
    - ....
  - When mathematical phrases are rendered in words:
    - equals~n,
    - less than~ϵ,
    - modulo~2,
    - for large~n,
    - ....
  - When cases are being enumerated within a paragraph:
    - Show that function $f(x)$ is (1)~continuous; (2)~bounded.

2.1.2 *Grouping*

Grouping is a common technique in LaTeX. The opening brace ({) starts a group and closing brace (}) closes it. Grouping has two purposes. The first purpose of grouping is that it turns several things into one compound thing. This may be needed, for example, if you want to pass several words to a command that typesets its argument in bold face text. The following demonstrates the point.

```
A bold \textbf{word} and
 a bold \textbf letter.
```

A bold **word** and a bold **l**etter.

The second purpose of grouping is that it lets you change certain settings and keep the changes local to the group. The following demonstrates how this may be used to make a local change to the type style of the text inside the group.

```
Normal text here.
{% Start a group.
 \bfseries
 % Now we have bold text.
 Bold paragraphs in here.
}% Close the group.
Back to normal text again.
```

Normal text here. **Bold paragraphs in here.** Back to normal text again.

Inside the group you may have several paragraphs. The advantage of the declaration `\bfseries` is that it defines how the text is typeset until the end of the group. The `\textbf` command just typesets its argument in a bold typeface. The argument may not contain paragraph-breaks.

There is also a low-level TeX mechanism for creating groups. It works just as the braces. A group is started with `\begingroup` and ended with `\endgroup`. These tokens may be freely mixed with braces but {/} pairs and `\begingroup`/`\endgroup` pairs should be properly matched. So `{ \begingroup \endgroup }` is allowed but `{ \begingroup } \endgroup` is not. A brace pair affects whitespace when you're typesetting mathematics but a `\begingroup`/`\endgroup` pair does not.

Table 2.2
Common diacritics

Output	Command	Name
ò	`\‘{o}`	Acute accent
ó	`\’{o}`	Grave accent
ô	`\^{o}`	Circumflex (hat)
õ	`\~{o}`	Tilde (squiggle)
ö	`\"{o}`	Umlaut or dieresis
ċ	`\.{c}`	Dot accent
š	`\v{s}`	Háček (caron or check)
ŏ	`\u{o}`	Breve accent
ō	`\={o}`	Macron (bar)
ő	`\H{o}`	Long Hungarian umlaut
o͡o	`\t{oo}`	Tie-after accent
ş	`\c{s}`	Cedilla accent
ọ	`\d{o}`	Dot-under accent
o̲	`\b{o}`	Bar-under accent

Table 2.3
Other special characters

Output	Command	Name
å	`\aa`	Scandinavian a-with-circle
Å	`\AA`	Scandinavian A-with-circle
ł	`\l`	Polish suppressed-l
Ł	`\L`	Polish suppressed-L
ø	`\o`	Scandinavian o-with-slash
Ø	`\O`	Scandinavian O-with-slash
¿	`?‘`	Open question mark
¡	`!‘`	Open exclamation mark

2.2 Diacritics

This section studies how to typeset characters with *diacritics*, which are also known as accents. Table 2.2 displays some commonly occurring diacritics and the commands that typeset them. The presentation is based on [Knuth 1990, Chapter 9].

Using `\"{i}` to typeset ï may not work if you're not using a Type 1 font (`T1` font). However, typesetting ï with `\"{\i}` should always work. Here the command `\i` is used to typeset a dotless i (ı). There is also a command `\j` for a dotless j.

Table 2.3 shows some other commonly occurring special characters.

2.3 Ligatures

A ligature combines two or several characters as a special glyph. Examples of English ligatures and their equivalent character combinations are fi (fi), ff (ff), ffi (ffi), fl (fl), and and ffl (ffl). LaTeX recognises English ligatures and substitutes them for the characters representing them.

Table 2.4 displays some foreign ligatures. The symbol ß (eszett) is

Table 2.4
Foreign ligatures

Output	Command	Name
œ	\oe	French ligature œ
Œ	\OE	French ligature Œ
æ	\ae	Scandinavian ligature æ
Æ	\AE	Scandinavian ligature Æ
ß	\ss	German 'Eszett' or sharp S

Figure 2.1
Quotes

```
'Convention' dictates that
punctuation go inside
quotes, like ``this,'' but
some think it's better
to do ``this''.
```

'Convention' dictates that punctuation go inside quotes, like "this," but some think it's better to do "this".

Figure 2.2
Nested quotations

```
``\,'Fi' or 'fum?'\,'' he asked.\\
```''Fi' or 'fum?''' he asked.  \\
``{}'Fi' or 'fum?'{}'' he asked.
```

"'Fi' or 'fum?'" he asked.
"'Fi' or 'fum?'" he asked.
"'Fi' or 'fum?'" he asked.

a ligature of ſs [Bringhurst 2008] and this is reflected in the LaTeX command that typesets the symbol.

Sometimes it is better to suppress ligatures. The following is an example: the \makebox command prevents LaTeX from turning the fi in selfish into a ligature, which makes the result much easier to parse: selfish, not selfish.

```
Mr~Crabs is a self\makebox{}ish shellfish.
```
LaTeX Usage

Other words that need "anti-hyphenation" pre-processing are halflife, halfline, selfless, offline, offloaded, and so on.

## 2.4 Quotation Marks

This section explains how you typeset quotation marks. Figure 2.1 is an example from [Lamport 1994, page 13]. The word 'Convention' in this example is in single quotes and the word 'this' is in double quotes. The quotes at the start are backquotes (` and ``). The quotes at the end are the usual quotes (' and ''). Notice that output quote between 'it' and 's' is produced using a single quote in LaTeX.

To get properly nested quotations you insert a thin space where the quotes "meet." Recall that the thin space command (\,) typesets a thin space. Figure 2.2 provides a concrete example that is taken from [Lamport 1994, page 14]. Figure 2.2 provides another example. The first line of this example looks much better than the other two. Note that LaTeX parses three consecutive quotes as a pair of quotes followed by one more quote. This is demonstrated by the second line of the output, which looks terrible. The last line of the input avoids the three consecutive quotes by adding an empty group that makes
```

explicit where the double quotes and the single quote meet. Still the resulting output doesn't look great.

Intermezzo. As a general rule, British usage prefers the use of single quotes for ordinary use. This poses a problem if an apostrophe is used for the possessive form: He said 'It is John's book.' This is why it is also acceptable to use double quotes [Trask 1997, Chapter 8].

2.5 Dashes

There are three kinds of dashes: -, –, and —. In LaTeX you get them by typing -, --, and ---. The second symbol can also be typeset with the command `\textendash` and the last symbol with the command `\textemdash`. The symbol –, which is used in mathematical expressions such as $a - b$, is not a dash. This symbol is discussed in Chapter 8. The following briefly explains how the dashes are used.

- This is the intra-word dash, which is used to hyphenate compound modifiers such as one-to-one, light-green, and so on [Trask 1997, Chapter 6]. In LaTeX you typeset this symbol as follows: -.
– This is the en-dash, which has the width of 1 en. An en is equivalent to half the *current* type size, so an en-dash is shorter in normal text than it is in large text. The en-dash is mainly used in ranges: pages 12–15 (from 12 to 15). However, the en-dash is also used to link two names that are sharing something in common: a joint Anglo–French venture [Allen 2001, page 45]. The LaTeX command `\textendash` and the sequence `--` typeset the en-dash. When you typeset an en-dash, it looks better if you add a little space before and after. Remember that `\,` produces a thin space. Use this command for the horizontal space.

LaTeX Usage
```
... pages~12\,--\,15 (from~12 to~15).
```

— This is the em-dash, which has the same width as an em. An em is equal to the *current* type size. The em-dash separates strong interruptions from the rest of the sentence—like this [Trask 1997, Chapter 6]. Bringhurst [2008, page 80] prefers the en-dash to the em-dash. The LaTeX command `\textemdash` and the sequence `---` typeset the em-dash. An em-dash at the start of a line doesn't look very good so you should tie each em-dash to the preceding word.

LaTeX Usage
```
... the rest of the sentence~\textemdash
like this~\parencite[Chapter~6]{Trask:1997}.
```

Figure 2.3 presents an example of the dashes. A few years ago I noticed that sometimes `---` doesn't work with XeTeX (even with `Mapping = tex-text` enabled). However, `\textemdash` always worked.

2.6 Full Stops

LaTeX usually treats a full stop (.) as an end-of-sentence indicator. By

```
The intra-word dash is used to hyphenate
 compound modifiers such as light-green,
 X-ray, or one-to-one. ...
The en-dash is used in ranges: pages~12--15.
The em-dash is used to separate strong
 interruptions from the rest of the
 sentence~--- like this%
 ~\cite[Chapter~6]{Trask:1997}. ...
```

Figure 2.3
Dashes

default, LaTeX inserts a bit more space after the full stop at the end of a sentence than it does between words. It also does this after other punctuation symbols. The `\frenchspacing` command turns this feature off. The command `\nonfrenchspacing` turns the feature on again. When a full stop is not the end of a sentence you need to help LaTeX a bit by inserting the space command (`\␣`) after the full stop.

LaTeX Usage
```
Meet me at 6~p.m.\␣at the Grand Parade.
```

However, when an uppercase letter is followed by a full stop, then LaTeX assumes the full stop is for abbreviation. For example:

LaTeX Usage
```
Donald~E. Knuth developed the {\TeX} system.
```

This convention causes a problem if an uppercase letter really is the end of a sentence. Insert a `\@` before the full stop if this happens.

LaTeX Usage
```
In Frank Herbert's \emph{Dune} saga,
 the Mother School of the Bene Gesserit
 is situated on the planet Wallach IX\@.
```

LaTeX inherits its habit of putting some extra space after full stops and other punctuation symbols from TeX. Bringhurst [2008, pages 28–30] points out that there really is no reason to add such extra space for modern works. Following Bringhurst's advice, this document was typeset with `\frenchspacing` enabled.

2.7 Ellipsis

The command `\ldots` produces an ellipsis (…), which is used to indicate an omission. If the ellipsis occurs at the end of a sentence, then you still need to add an end-of-sentence marking full stop. If this happens then Felici [2012, Figure 13.15] recommends that you put the ellipsis close to the preceding text and then add the full stop.

```
Many stories start with
 'Once upon a time\ldots'
They usually end with
 '\ldots\␣and they all lived
  happily ever after.'
```

Many stories start with 'Once upon a time….' They usually end with '… and they all lived happily ever after.'

Figure 2.4
Good borderline punctuation

```
Robert Bringhurst, author of
 \emph{Elements of
       Typographic Style,}
 recommends setting such
 punctuation symbols in
 the brighter type.
\textbf{Do as he}, or
 risk getting ugly type.
```

Robert Bringhurst, author of *Elements of Typographic Style,* recommends setting such punctuation symbols in the brighter type. **Do as he**, or risk getting ugly type.

Figure 2.5
Poor borderline punctuation

```
Robert Bringhurst, author of
 \emph{Elements of
       Typographic Style},
 recommends setting such
 punctuation symbols in
 the brighter type.
\textbf{Do as he,} or
 risk getting ugly type.
```

Robert Bringhurst, author of *Elements of Typographic Style*, recommends setting such punctuation symbols in the brighter type. **Do as he,** or risk getting ugly type.

2.8 Emphasis

Emphasis is a typographic tool for typesetting text in a different typeface. The idea is that this makes the text stand out. Emphasis is especially useful when introducing a new concept, such as in this paragraph.

In some documents, emphasis is implemented by typesetting text in a bold face typeface, by typesetting it in uppercase typeface, or (worse) by underlining the text. LaTeX emphasises text in paragraphs by italicising the text. Trask [1997, page 82] calls this the preferred style for emphasis. The LaTeX command for emphasis is `\emph`.

```
Emphasised \emph{example}.
```

Emphasised *example*.

2.9 Borderline Punctuation

Bold text looks darker than normal, upright text and italicised text look brighter than normal, upright text. When small punctuation symbols get caught between darker and brighter type it is time to pay attention. Robert Bringhurst, author of *Elements of Typographic Style,* recommends setting such punctuation symbols in the brighter type [Bringhurst 2008]. **Do as he**, or risk getting ugly type. Figures 2.4 and 2.5 demonstrate what you get if you follow Bringhurst's advice and what if you don't. The figures do not excel in terms of maintainability because they hardcode the author's name and the title of the work.

2.10 Footnotes and Marginal Notes

It is generally accepted that using footnotes and marginal notes should be used sparingly because they are disruptive. However, proper use of

```
Footnotes\footnote{A footnote is a note
 of reference, explanation, or comment that is
 usually placed below the text on a printed page.}
 can be a nuisance. This is especially true if
 there are many.\footnote{Like here.} The more you see
 them, the more annoying they get.\footnote{Got it?}
```

Figure 2.6
Using footnotes

Footnotes[a] can be a nuisance. This is especially true if there are many.[b] The more you see them, the more annoying they get.[c]

[a] A footnote is a note of reference, explanation, or comment that is usually placed below the text on a printed page.
[b] Like here.
[c] Got it?

marginal notes in documents with wide margins can be very effective.

Not surprisingly, LaTeX provides a command for footnotes and a command for marginal notes. Figure 2.6 demonstrates how to specify footnotes in LaTeX. A *marginal note* or *marginal paragraph* is like a footnote, but placed in the margin as on this page. The command `\marginpar{⟨text⟩}` puts ⟨text⟩ in the margin as a marginal note. By passing an optional argument to the command you can put different text on odd (recto/front/right) pages and on even (verso/back/left) pages. The optional argument is used for even pages and the required argument is used for odd pages. If you're using both the optional and required argument then it is easy to remember which is which: the optional argument is to the left of the required argument so it's for the left page; the required argument is for the right page. Note that narrow marginal notes may look better with ragged text, which is text that is aligned to one side only. On the right (left) pages you use ragged right (left) text. Section 2.19.2 explains how to typeset ragged text.

Avoid marginal notes in very narrow margins.

2.11 Displayed Quotations and Verses

The `quote` and `quotation` environments are for typesetting displayed quotations. The former is for short quotations; the latter is for longer quotations. Figure 2.7 shows how you use the `quote` environment. The command `\\` in Figure 2.7 forces a line break.

The `verse` environment typesets poetry and verse. Figure 2.8 shows how you use the environment. In this example, the command `\qquad` inserts two quads. Here a quad is an amount of space that is equivalent to the *current* type size. So if you use a 12 pt typeface then a quad results in a 12 pt space in normal text. The command `\\` inside the `verse` environment determines the line breaks. Remember that the command `\,` before the letter *S* inserts a thin space.

2.12 Line Breaks

In the previous section, the command `\\` inserted a line break in displayed quotations and verses. The command also works inside

Figure 2.7 The quote environment

```
Blah blah blah blah blah blah blah blah blah blah blah.
\begin{quote}
   Next to the originator of a good sentence
    is the first quoter of it. \\
   \emph{Ralph Waldo Emerson}
\end{quote}
Blah blah blah blah blah blah blah blah blah blah blah.
```

Blah blah blah blah blah blah blah blah blah blah blah.

> Next to the originator of a good sentence is the first quoter of it.
> *Ralph Waldo Emerson*

Blah blah blah blah blah blah blah blah blah blah blah.

Figure 2.8 The verse environment

```
The following anti-limerick is
 attributed to W.\,S. Gilbert.
\begin{verse}
  There was an old man of St.~Bees,            \\
  Who was stung in the arm by a wasp;          \\
    \qquad When they asked, ''Does it hurt?''  \\
    \qquad He replied, ''No, it does n't,      \\
  But I thought all the while 't was a Hornet.''
\end{verse}
```

The following anti-limerick is attributed to W. S. Gilbert.

> There was an old man of St. Bees,
> Who was stung in the arm by a wasp;
> When they asked, "Does it hurt?"
> He replied, "No, it does n't,
> But I thought all the while 't was a Hornet."

paragraphs. An optional argument determines the extra vertical space of the line break: `\\[⟨extra vertical space⟩]`. A line break at the end of a page may trigger a page break. If you don't want a page break then you should use the command `\\*`. It is identical to `\\` but it inhibits page breaks.

2.13 Controlling the Size

With the proper class and packages there is usually no need to change the type size of your text. However, sometimes it has its merits, e.g., when you're designing your own titlepage or environment. Table 2.5 lists the declarations and environment that change the type size. The preferred "size" for long-ish algorithms and program listings is `\scriptsize`. If you're using a package to typeset listings then the package usually chooses the right size. If not, it probably lets you specify the type size. Figure 2.9 shows how you change the size of text.

Declaration	Environment	Example
\tiny	tiny	Example
\scriptsize	scriptsize	Example
\footnotesize	footnotesize	Example
\small	small	Example
\normalsize	normalsize	Example
\large	large	Example
\Large	Large	Example
\LARGE	LARGE	Example
\huge	huge	Example
\Huge	Huge	Example

Table 2.5
Size-affecting declarations and environments

```
{\tiny Mumble.      \\
 \begin{normalsize}
      What?
 \end{normalsize}   \\
 \begin{Huge}
      Mumble!
 \end{Huge} }
```

Mumble.
What?
Mumble!

Figure 2.9
Controlling the size

2.14 Seriffed and Sans Serif Typefaces

LaTeX has several commands that change the type style. Before studying these commands it is useful to study the difference between seriffed and sans serif typefaces and when to use them.

A *serif* is a little decoration at the end of some of the strokes of some of the letters. In a *seriffed* typeface the letters have serifs. Seriffed typefaces are sometimes called *roman* typefaces but in LaTeX roman means upright. In a *sans serif* typeface the letters lack serifs.

Most books use a seriffed typeface for the running text [Unger 2007, pp. 167–168] and the most popular typeface for the running text of books and reports is *(Monotype/Linotype) Times Roman* [Felici 2012], a seriffed typeface. Seriffed typefaces are also used for the running text of most papers, theses, and dissertations in science. Turabian [2007, pp. 374–375] recommends that you use a typeface that is designed for text and that you use a size in the range of 10–12 pt, with 12 pt being the preferred size. Admittedly, the being designed for text is a bit vague but Turabian [2007] give two examples, both of which are seriffed.

As lines get longer and longer, seriffed typefaces are easier to read and make fast reading easier [Unger 2007]. Sans serif typefaces may look better on the screen but the ultimate criterion for printed matter is how the text looks in print, so never choose the typeface for your printed text based on how it looks on the screen.

If a typeface family has a seriffed and sans serif typeface of the same

type size (point size), then the seriffed typeface usually requires more horizontal space [Unger 2007]. Stated differently, sans serif typefaces are usually more efficient when it comes to saving space. This may be exploited by using sans serif typefaces in captions, in brochures, in short narrow columns, or on road signs [Unger 2007].

If you don't change the typeface then LaTeX will typeset the body of your document in *Computer Modern*. An example of *Computer Modern* may be found in Table 2.6, further on in this chapter.

2.15 Small Caps Letters

Small caps letters are used to typeset acronyms and abbreviations. Their shape is the same as uppercase letter but their height is smaller, which lets them blend in better with the rest of the text. For example, compare NO SHOUTING with no shouting. The latter is easier on the eye.

Adding extra space uniformly to the left and right of characters in a passage of text is called tracking or letterspacing. The extra space that is added per letter is called the tracking space. Tracking passages of small caps text is a common technique to improve the legibility. For example non-spaced small caps is not spaced, whereas spaced small caps is letterspaced.

The command `\textsc` typesets lowercase letters in small caps. The easiest way to automatically letterspace such text is to use the `microtype` package with the option `tracking=smallcaps`. After this all small caps text will be letterspaced.

```
\textsc{No shouting}.          No shouting.
```

The `microtype` package also provides character protrusion (margin kerning) and font expansion. Character protrusion adjusts the characters at the margins of the text. Font expansion uses narrow or wider font versions so as to make the overall appearance of the text more uniform, avoiding long cramped, dark lines with many characters and long loose, bright lines with few characters. As a side-effect, font expansion may also be used to choose better hyphenation points [Schlicht 2010]. This document was typeset using the `microtype` package with the following options.

LaTeX Usage
```
\usepackage[final,tracking=smallcaps,
            expansion=alltext,protrusion=true]{microtype}
```

Bringhurst [2008, page 30] recommends that you add 5–10% of the type size (point size) for the tracking space. The `microtype` package expects the extra tracking in thousands of the type size. The following sets the tracking space to 5% for the `sc` (small caps) shape.

LaTeX Usage
```
\SetTracking{encoding=*,shape=sc}{50}
```

Most `microtype` users agree that the package improves the appearance of their documents.

Declaration	Command	Example
\mdseries	\textmd	Medium Series
\normalfont	\textnormal	Normal Style
\rmfamily	\textrm	Roman family
\upshape	\textup	Upright Shape
\itshape	\textit	*Italic Shape*
\slshape	\textsl	*Slanted Shape*
\bfseries	\textbf	**Boldface Series**
\scshape	\textsc	Small Caps Shape
\sffamily	\textsf	Sans Serif Family
\ttfamily	\texttt	`Typewriter Family`

Table 2.6
Type style affecting declarations and commands. The last column shows the result in *Computer Modern* (LaTeX's default typeface). The first four lines usually correspond to the default style. The first nine typefaces are proportional. They may have glyphs with different widths, e.g., compare *M* and *i*. Small caps letters are useful for abbreviations. The last typeface is non-proportional, which is useful in program listings.

2.16 Controlling the Type Style

Changing the type size is hardly ever needed in an article, thesis, report, or book. Changing the type style is required much more, but usually this is done automatically by the commands that typeset the title of your document, the section titles, the captions, and so on.

There are ten LaTeX type style affecting declarations. Each declaration has a command that takes an argument and applies the type style of the declaration to the argument. The arguments cannot have paragraph breaks. The declarations and commands are listed in Table 2.6.

Intermezzo. If you really must change the type style of your text then it is probably for a specific purpose. For example, to change the type style of a newly defined word, to change the type style of an identifier in an algorithm, and so on. Rather than hard-coding the style in your input, it is better if you define a user-defined command that typesets your text in the required style and use the command to typeset your text. The command's name should reflect its purpose. For example `\identifier` to typeset an identifier in an algorithm, `\package` to typeset the name of a LaTeX package, and so on. Using this approach improves maintainability. For example, if you want to change the type style of all identifiers in your text then you only need to make changes in the definition of the command that typesets identifiers. Defining your own commands is discussed in Chapter 11.

2.17 Abbreviations

This section is about abbreviations. It provides some guidelines about their spelling and how to typeset them in LaTeX.

2.17.1 *Initialisms*

Abbreviations that are made up of the initial letters of the abbreviated words are called *initialisms*. Non-standard initialism are usually written with a full stop after each part in the abbreviation: Ph.D. (Philosophiae

Figure 2.10 Finer points of typesetting abbreviations

```
In~2010 Prof.~Donald Knuth was invited to the
 annual TeX User Group Conference in San Francisco, Ca.\␣
 to speak about a revolutionary successor to \TeX.
This remarkable system is entirely menu driven and
 incorporates facilities for social networking.
Pronouncing the name involves making the sound of a bell.
```

Doctor), D.Phil. (Doctor of Philosophy), M.Sc. (Master of Science), and so on. However, if the initialisms are standard, then you omit the full stops, so B.B.C. becomes BBC, 4 G.L. (fourth-Generation Language) becomes 4GL, and Ph.D. becomes Ph D (in LaTeX `Ph~D`). Bringhurst [2008, page 48] recommends typesetting abbreviations with more than two uppercase letters in spaced small capitals: SPACED SMALL CAPS. Section 2.15 explains how to get spaced small caps.

Some authors recommend that you letterspace Uniform Resource Locators (URLs), phone numbers, and email addresses because they are not words. See for example [Bringhurst 2008] or [Hedrick 2003].

Abbreviations of personal names such as D. E. K., J. F. K., J. S. B., and the like should not be letterspaced.

2.17.2 *Acronyms*

An *acronym* is an initialism that is pronounced as a word. For example, radar (RAdio Detection And Ranging), sonar (SOund Navigation And Ranging), NASA (National Aeronautics and Space Administration), and EBCDIC (Extended Binary Coded Decimal Interchange Code); but not ACM (Association for Computing Machinery), BBC (British Broadcasting Corporation), and RSVP (Répondez S'il Vous Plaît). Note that not all acronyms are spelt with uppercase letters; if you're not certain, look up the spelling. Since acronyms are just a special form of initialisms, we should follow Bringhurst's advice, and write them with small caps if they are spelt with (two or more) uppercase letters.

2.17.3 *Shortenings*

A word that is abbreviated by taking the first few letters of that word is called a *shortening*. To avoid ambiguity, shortenings are usually written with a full stop at the end of each part. For example, p. (page), proc. (proceedings), sym. (symposium), fig. (figure), Feb. (February), Prof. (Professor), and so on. The abbreviation pp. is for pages.

Remember that LaTeX inserts a little extra white space after a full stop if `\frencspacing` isn't enabled. If an abbreviation is not at the end of a sentence and ends with a full stop then this extra space may look bad. To suppress the extra white space you have to hardcode a space command (`\␣`) after the abbreviation or tie the abbreviation and the following word. Figure 2.10 provides a small example.

2.17.4 *Introducing Abbreviations*

The first time you introduce an abbreviation you should explain it. Most authors first spell out the abbreviation and then provide the abbreviation in parenthesis. The `acronym` package provides some support for defining and referencing abbreviations in a consistent style. This is done using the standard label-referencing technique. The package provides commands for singular and plural versions of abbreviations and for abbreviated and unabbreviated versions.

Page 4 of this book introduces an acronym for integrated development environments. This text was generated by the following input.

LaTeX Input
```
... many \acp{IDE} ...
```

The command `\acp` in this example is provided by the `acronym` package. The command introduces the plural version of an abbreviation. The `acronym` package also provides the `\ac` command, which introduces the singular version of an abbreviation. The argument `IDE` of the `\acp` command is the label of the acronym. Some other part of the input associates the label `IDE` with the abbreviated version 'IDE' and the expanded version 'Integrated Development Environment.' This was (essentially) done as follows:

LaTeX Input
```
\acro{IDE}[\textsc{ide}]%
     {Integrated Development Environment}
```

When this book was generated and the command `\acp` was used in the second last input, this was the first time the label `IDE` was referenced, which is why it resulted in the following output.

LaTeX Output
```
... many Integrated Development Environments (IDEs) ...
```

The label `IDE` is also referenced in other locations in the input, but when that happens it always results in the abbreviated version of the acronym: IDE. More information about the `acronym` package may be found in the package documentation [Oetiker 2010].

2.17.5 *British and American Spelling*

There are differences between American and British usage in time abbreviations. According to Trask [1997] Americans write 10:05 AM (Ante Meridiem) for five past ten in the morning and 13:15 PM (Post Meridiem) for a quarter past one in the afternoon. British spelling prefers 10.05 a.m. and 13.15 p.m. [Trask 1997]. Felici [2012] notices that Americans have also started using the British form.

For titles such as Mister, Doctor, and so on, British and American usage differ. Britsh usage is the same as for shortenings. For example, Mr Happy, Dr Who, and Fr Dougal McGuire. Americans add the full stop: Mr. Ed, Dr. Quinn, Medicine Woman, and Fr. Bob Maguire.

Table 2.7
Latin abbreviations. The first column lists the abbreviations, the second the original Latin meaning, and the last the English translation. Note that the abbreviations at the bottom of the table are slanted. This is intensional and preferred usage.

Abbreviation	Latin meaning	English meaning
e.g.	exempli gratia	for example
i.e.	id est	that is/in other words
etc.	et cetera	and so forth
viz.	videlicet	that is to say/namely
cf.	confer	compare
et al.	et aluis	and others

2.17.6 *Latin Abbreviations*

This section studies some Latin abbreviations that are commonly used in scientific writing. Table 2.7 presents the more commonly occurring abbreviations, their Latin meaning, and the English translation.

Note that some abbreviations are typeset in italics. This is not by accident: this is how they should be typeset—but conventions may differ from field to field. Also note that the *al* in *et al.* gets a full stop because it is an abbreviation of *aluis* but that the *et* does not get a full stop because it is already spelt out in full. Remember Bringhurst's advice and put the full stop inside the argument of `\emph`: `\emph{et al.}` Finally note that etc. is short for et cetera, not for ectcetra.

Trask [1997] discourages these abbreviations. Trask continues by pointing out that writing statements like the following are wrong because the reader should be invited to consult the reference.

> The Australian language Dyirbal has a remarkable gender system, *cf.* [Dixon 1972].

Trask proposes the following solution.

> The Australian language Dyirbal has a remarkable gender system; see [Dixon 1972].

Abbreviations such as etc., i.e., and e.g. require additional punctuation [Strunk, and White 2000]:

- Abbreviations such as BBC, NBC, etc., are called initialisms.
- Shortenings, i.e., abbreviations that are formed by taking the first letters of the abbreviated word, usually end with a full stop.
- Abbreviations are not always spelt te same, e.g., Ph.D. and Ph D.

2.17.7 *Units*

The Système International d'Unités/International System of Units (SI) provides rules for consistent typesetting of quantities of units. Heldoorn [2007] provides a summary of these rules. The following is a summary of the main rules.

- The base unit symbols are printed in upright roman: g (gram), m (metre), t (tonne), ... Exceptions are unit symbols that are spelt in Greek and the symbols for inch, degrees, seconds, and so on.

```
Fill in the missing word.\\
Fill in the missing
 \phantom{word}.
```

Fill in the missing word.
Fill in the missing .

Figure 2.11
The `\phantom` command

- The first letter of the unit symbol is uppercase if it is derived from a proper name: Å (Ångström), N (Newton), Pa (Pascal), ….
- The plural form of the base unit symbol is the same as the singular.
- The base unit symbols do not receive an end-of-abbreviation full stop.

Needless to say, it is important that you typeset quantities of units correctly and consistently. The hard way is doing it by hand. The easy way is doing it with LaTeX.

At the moment of writing the most popular package for specifying SI units is the `siunitx` package [Wright 2011].

- It provides support to configure how the SI units are typeset. For example, kg m s^{-1}, versus kg m s^{-1}, versus kg m/s, and so on.
- It provides commands to typeset quantities of units: `\SI[mode=text]{1.23}{\kilogram}` will give you 1.23 kg and `\SI{1.01}{\kilogram}` will typeset 1.01 kg in the default typesetting mode.
- The package provides macros to typeset lists of quantities in a given unit. For example `\SIlist{0.1;0.2;1.0}{\milli\metre}` gives you 0.1, 0.2 and 1.0 mm if the default typesetting mode is `text`. If you add the option `list-final-separator={, and~}` then you get 0.1, 0.2, and 1.0 mm.
- By default, unit symbols are typeset using the default math roman font but you can also use different fonts.

Discussing the entire `siunitx` package is beyond the scope of this book. The interested reader is referred to the package documentation [Wright 2011] for further information.

2.18 Phantom Text

Some commands don't typeset anything with ink but do affect the horizontal and vertical spacing. The following is the first of three useful versions.

`\phantom{`⟨`stuff`⟩`}`
This command "typesets" its argument using invisible ink. The dimensions of the box are the same as the dimensions required for typesetting ⟨`stuff`⟩. ☑

Figure 2.11 demonstrates how you use the command. The `\hphantom` and `\vphantom` commands are horizontal and vertical versions of the `\phantom` command. The following explains how they work.

`\hphantom{`⟨`stuff`⟩`}`
This is the horizontal version of the `\phantom` command. The command creates a box with zero height and the same width as its argument, ⟨`stuff`⟩. ☑

Figure 2.12
The center environment

```
\begin{center}
  Blah.\\
  Blah blah blah.

  Blah blah blah blah blah
  blah blah blah blah blah
  blah blah blah blah blah.
\end{center}
```

Blah.
Blah blah blah.
Blah blah blah blah blah blah
blah blah blah blah blah blah
blah blah blah.

\vphantom{⟨stuff⟩}
This is the vertical version of the \phantom command. The command creates a box with zero width and the same height as its argument, ⟨stuff⟩. It is especially useful for getting the right size for delimiters such as parentheses in mathematical formulae that span multiple lines. This is explained in more detail in Section 8.8.1. ☑

2.19 Alignment

This section studies three commands and two environments that change the text alignment. The first command centres text. The second and third command align text to the left and to the right. The first of the environments is the tabular environment, which typesets row-based content with horizontal alignment positions (columns). The last environment is the tabbing environment. This environment lets you define horizontal alignment (tab) positions and lets you position text relative to these alignment positions.

2.19.1 *Centred Text*

The center environment centres text. The example in Figure 2.12 demonstrates the environment. The example is inspired by Iggy Pop.

2.19.2 *Flushed/Ragged Text*

The flushleft environment and the \raggedright declaration typeset text that is aligned to the left. Likewise, the flushright environment and \raggedleft declaration typeset text that is aligned to the right. The example in Figure 2.13 shows the effect of the flushleft environment.

2.19.3 *Basic tabular Constructs*

The tabular environment typesets text with rows and alignment positions for columns. The environment also has siblings called tabular* and array. The tabular* environment works similar to tabular but it takes an additional argument that determines the width of the resulting construct. This environment is explained in Section 2.19.5. The

```
\begin{flushleft}
  Blah.\\
  Blah blah blah.

  Blah blah blah blah blah
  blah blah blah blah blah
  blah blah blah blah blah.
\end{flushleft}
```

Blah.
Blah blah blah.
Blah blah blah blah blah blah blah blah blah blah blah blah blah blah.

Figure 2.13
The `flushleft` environment

`array` environment can only be used in math mode. The `tabular` and `tabular*` environments can be used in both text and math mode.

The remainder of this section introduces the `tabular` environment. This introduction should more than likely suffice for day-to-day usage. A more detailed presentation is provided in Section 2.19.5.

In its simplest form the `tabular` environment is used as follows.

LaTeX Usage

```
\begin{tabular}[⟨global alignment⟩]
               {⟨column alignment⟩}
   ⟨text⟩ & ⟨text⟩ & ... & ⟨text⟩ \\
   ...
   ⟨text⟩ & ⟨text⟩ & ... & ⟨text⟩ \\
   ⟨text⟩ & ⟨text⟩ & ... & ⟨text⟩
\end{tabular}
```

The body of the environment contains a sequence of rows that are delimited by linebreaks (`\\`). Each row is a sequence of alignments tab-delimited ⟨`text`⟩. The *i*-th ⟨`text`⟩ in a row corresponds to the *i*-th column. The following explains the arguments of the environment:

⟨**global alignment**⟩
This optional argument determines the vertical alignment of the environment. Allowed values are `t` (align on the top row), `c` (align on the centre), or `b` (align on the bottom row). The default value of this argument is `c`. √

⟨**column alignment**⟩
This argument determines the column alignment and additional decorations. For day-to-day usage, the following options are relevant.

- **`l`** This option corresponds to a left-aligned column.
- **`r`** This corresponds to a right-aligned column.
- **`c`** This corresponds to a centred column.
- **`p{`**⟨**`width`**⟩**`}`** This option corresponds to a top-aligned ⟨`width`⟩-wide column that is typeset as a paragraph in the "usual" way. Some commands such as `\\` are not allowed at the top level.
- **`|`** This option does not correspond to an actual column but results in additional decoration. It results in a vertical line drawn at at the "current" position. For example, if ⟨`column alignment`⟩ is `l|cr` then there will be a vertical line separating the first two columns. Using this option is discouraged because the vertical lines usually distract. √

The `tabular` environment also defines the following commands,

Figure 2.14 Using the `tabular` environment

```
\begin{tabular}{l|crp{3.1cm}}
  \hline
     1 &   2 &   3
       & Box me in,
         but not too
         tight, please.
\\\hline
    11 &  12 &  13 & Excellent.
\\ 111 & 112 & 113 & Thank you!
\\\hline
\end{tabular}
```

1	2	3	Box me in, but not too tight, please.
11	12	13	Excellent.
111	112	113	Thank you!

which may be used inside the environment. You can only use these commands at the start of a row.

\hline

This command inserts a horizontal rule. The command may only be used at the *start* of a row. ☑

\cline{⟨number$_1$⟩}{⟨number$_2$⟩}

This draws a horizontal line from the start of column ⟨number$_1$⟩ to the end of column ⟨number$_2$⟩. ☑

\vline

This results in a vertical line. The command may only be used if the column is aligned to the left, to the right, or to the centre. ☑

Figure 2.14 presents a simple example of the `tabular` environment. The example shows all alignments and the paragraph feature.

Note that line breaks are inserted automatically inside `p`-type columns. Line breaks are not allowed in columns aligned with `l`, `r`, or `c`.

Intermezzo. The column alignment option `|` and the commands `\hline`, `\cline`, and `\vline` are irresistible to new users. This may be because most examples of the `tabular` environment involve the option and these commands. It is understandable that new users want to repeat this, especially when they're not aware that using the option and the commands in moderation is better because the grid lines are dazzling and distracting. Chapter 6 provides some guidelines on how to design good tables.

Regular $m \times n$ tables with the same alignment in the same column are rare. The following command lets you join columns within a row and override the default alignment.

\multicolumn{⟨number⟩}{⟨column alignment⟩}{⟨text⟩}

This inserts ⟨text⟩ into a *single* column that is formed by combining the next ⟨number⟩ columns in the current row. The alignment of the column is determined by ⟨column alignment⟩. This command is especially useful for overriding the default alignment in column headings of a table. An example is presented in the next section. ☑

2.19.4 *The booktabs Package*

The booktabs package adds some extra functionality to the tabular environment. The package discourages vertical grid lines. Using the booktabs package results in better looking tables.

- The package provides different commands for different rules.
- The package provides different rules that may have different widths.
- The package provides commands for temporarily/permanently changing the width.
- The package has a command that adds extra line space.
- The package is compatible-ish with the colortbl package, which is used to specify coloured tables.

The booktabs package provides the following commands. The first four commands take an option that specifies the width of the rule. The first four commands can only be used at the start of a row.

\toprule[⟨width⟩]
This typesets the full horizontal rule at the top the table. √

\bottomrule[⟨width⟩]
This typesets the full horizontal rule at the bottom of the table. √

\midrule[⟨width⟩]
This typesets the remaining full horizontal rules in the table. √

\cmidrule[⟨width⟩]{⟨number$_1$⟩-⟨number$_2$⟩}
This typesets a partial horizontal rule. The rule is supposed to be used in the middle of the table. It ranges from the start of column ⟨number$_1$⟩ to the end of column ⟨number$_2$⟩. √

\addlinespace[⟨height⟩]
This command is usually used immediately after a line break and it inserts more vertical line space to the height of ⟨height⟩. √

Figure 2.15 demonstrates how to use the booktabs-provided rule commands. The resulting output is presented in Figure 2.16. Notice that the inter-linespacing is much better than the output in Figure 2.14. Also notice the different widths of the rules.

2.19.5 *Advanced tabular Constructs*

Using basic tabular constructs usually suffices for day-to-day typesetting. This section explains the techniques that give you the power to typeset more advanced tabular constructs.

The following starts by presenting two addition column options. This is followed by some style parameters that control the default size and spacing of the tabular, tabular*, and array environments. The column options are as follows.

***{⟨number⟩}{⟨column options⟩}**
This inserts ⟨number⟩ copies of ⟨column options⟩. For example, *{2}{lr} is equivalent to lrlr. √

@{⟨text⟩}
This is called an @-expression. It inserts ⟨text⟩ at the current position.

Figure 2.15
Input of `booktabs` package

```
\begin{tabular}[c]{lrrp{47mm}}
  \toprule \multicolumn{1}{r}{\textbf{Destination}}
         & \multicolumn{1}{r}{\textbf{Duration}}
         & \multicolumn{1}{r}{\textbf{Price}}
         & \multicolumn{1}{r}{\textbf{Description}}
\\\midrule
   Cork City
   &  7 Days & \euro 300.00
   & Visit Langer Land. Price includes visits
     to Rory Gallagher Place and de Maarkit.
\\ Dingle
   &  8 Days & \euro 400.00
   & Have fun with Fungie.
\\\bottomrule
\end{tabular}
```

Figure 2.16
Output of `booktabs` package. The input of this figure is listed in Figure 2.15. Clearly, `booktabs` rules rule.

Destination	Duration	Price	Description
Cork City	7 Days	€300.00	Visit Langer Land. Price includes visits to Rory Gallagher Place and de Maarkit.
Dingle	8 Days	€400.00	Have fun with Fungie.

This is useful if you want to add certain text or symbols at the given position. For example `@{.}` inserts a full stop at the current position.

LaTeX normally inserts some horizontal space before the first column and after the last column. It inserts twice that amount of space between adjacent columns. However, this space is suppressed if an @-expression precedes or follows a column option. For example, if ⟨column alignment⟩ is equal to `@{}ll@{}l@{}` then this suppresses the horizontal space before the first column, after the last column, and between the second and last column. The length `\tabcolsep` controls the extra horzontal space that is inserted. The value of the command is half the width that is inserted between columns.

A horizontal spacing command in an @-expression controls the separation of two adjacent columns. For example, `@{\hspace{⟨width⟩}}` inserts a horizontal ⟨width⟩-wide space.

Finally, @-expressions may also adjust the default column separation. The `\extracolsep{⟨width⟩}` adds additional horizontal ⟨width⟩-wide space between subsequent columns. *However,* additional width is never inserted before the first column. The `\extracolsep{\fill}` inserts the maximum possible amount of horizontal space. This is useful if you want to extend the width to the maximum possible width.

The columns in the second table in Figure 2.17 are spread out evenly with an @-expression. The third table adds the usual space to the start of the first and the end of the last column. The first table is added for comparison. ☑

```
\begin{tabular*}{3cm}{@{}lcr@{}}
  \toprule M & M & M \\\bottomrule
\end{tabular*}
\begin{tabular*}{3cm}
      {@{\extracolsep{\fill}}%
       lcr%
       @{\hspace{0pt}}}
  \toprule M & M & M \\\bottomrule
\end{tabular*}
\begin{tabular*}{3cm}
      {@{\hspace{\tabcolsep}}%
       @{\extracolsep{\fill}}%
       lcr%
       @{\hspace{\tabcolsep}}}
  \toprule M & M & M \\\bottomrule
\end{tabular*}
```

M	M	M

M	M	M

M	M	M

Figure 2.17
Controlling column widths with an @-expression. The output is spaced out for clarity.

The following commands control the default appearance of `tabular`, `tabular*`, and `array` environments.

\arraycolsep
The value of this length command is equal to half the default horizontal distance between adjacent columns in the `array` environment. This amount of space is also equal to the default horizontal space inserted before the first column and after the last column. √

\tabcolsep
The value of this length command is equal to half the default horizontal distance between adjacent columns in the `tabular` and `tabular*` environments. Again, this is equal to the default horizontal space that is inserted before the first column and after the last column. √

\arrayrulewidth
The value of this length command is the width of the lines resulting from a `|` in the ⟨`column options`⟩ argument and the lines resulting from the commands `\cline`, `\hline`, and `\vline`. √

\doublerulesep
The value of this length command is the distance between two adjacent lines resulting from a `||` in the ⟨`column options`⟩ argument or two adjacent lines resulting from the `\hline` command. √

\arraystretch
This command determines the distance between successive rows. It defaults to 1 and "multiplying" it by *x* results in rows that are *x* times further apart. So, by redefining this command to 0.50 you halve the row distance. Redefining commands is explained in Chapter 11. √

2.19.6 *The `tabbing` Environment*

The `tabbing` environment is useful for positioning material relative to user-defined alignment positions. The remainder of this section

Figure 2.18
The tabbing environment

```
\begin{tabbing}
From \=here to \=there \\
   \>and \>then\\
   \>\>all\\
   \>the \>way\\
back \>to \>here.
\end{tabbing}
```

From here to there
and then
all
the way
back to here.

Figure 2.19
Advanced tabbing

```
\begin{tt}\begin{tabbing}
AAA\=AAA\=AAA\=AAA \kill
FUNC euc( INT a,
          INT b ): INT \\
BEGIN \+ \\
  WHILE (b != 0) DO \\
  BEGIN \+ \\
    INT rem = a MOD b;\\
    a = b; \\
    b = rem; \- \\
  END \\
  RETURN a; \- \\
END;
\end{tabbing}\end{tt}
```

```
FUNC euc( INT a, INT b ): INT
BEGIN
   WHILE (b != 0) DO
   BEGIN
      INT rem = a MOD b;
      a = b;
      b = rem;
   END
   RETURN a;
END;
```

describes some basic usage of the environment. The reader is referred to [Lamport 1994, pages 201–203] for more detailed information.

The `tabbing` environment can only be used in *paragraph mode* (the "usual mode"). It produces lines of text with alignment in columns based upon *tab positions*.

\=
Defines the next tab (alignment) position. ☑

****
Inserts a line break and resets the next tab position to the value of `left_margin_tab`. ☑

\kill
Throws away the current line but remembers the tab positions defined with \=. ☑

\+
Increments `left_margin_tab`. ☑

\-
Decrements `left_margin_tab`. ☑

\>
Move to the next tab stop. ☑

Figures 2.18 and 2.19 present two examples of the `tabbing` environment. The examples do not demonstrate the full functionality of the environment.

```
\usepackage[dutch,british]{babel}
⋮
\selectlanguage{dutch}
% Dutch text here.
 Nederlandse tekst hier.

\selectlanguage{british}
% Engelse tekst hier.
 English text here.
```

Figure 2.20
Using the `babel` package

2.20 Language Related Issues

As suggested by its title, this section is concerned with language related issues. The remaining three sections deal with hyphenation, foreign languages, and spelling.

2.20.1 *Hyphenation*

LaTeX's (TeX's really) automatic hyphenation is second to none. However, sometimes even TeX gets it wrong. There are two ways to overcome such problems.

- The command `\-` in a word tells LaTeX that it may hyphenate the word at that position.

```
Er\-go\-no\-mic has three hyphenation positions.
```
LaTeX Usage

- Specifying the same hyphenation patterns is messy and prone to errors. Using the `\hyphenation` command is a much cleaner solution. This command takes one argument, which should be a comma-separated list of words. For each word you can put a hyphen at the (only) possible, desired, or allowed hyphenation positions. You may use the command several times. The following is an example.

```
\hyphenation{fortran,er-go-no-mic}
```
LaTeX Usage

2.20.2 *Foreign Languages*

The `babel` package supports multi-lingual documents. The package supports proper hyphenation, switches between different languages in one single document, definition of foreign languages, commands that recognise the "current" language, and so on. Figure 2.20 provides a minimal example. Rik Kabel kindly informed that XeTeX users use the `polyglossia` package instead of `babel`. One of the advantages of the `polyglossia` package is that it automatically loads the `bidi` package when bi-directional scripts are used.

2.20.3 *Spell-Checking*

LaTeX does not support automatic spell-checking. Note that spell-checking isn't trivial anyway because commands may generate text. Text may come from external files, so make sure you spell-check your bibliography files.

However, most modern IDEs have a spell checker. The `ispell` program, which can be run from the command line, has a LaTeX spell-check mode. The `-t` flag tells the command that the input is LaTeX.

Unix Session
```
$ ispell -l -t -S input.tex | sort -u
```