1 Typography

1.1 Introduction

Throughout the millennia humans have developed and adapted methods for storing facts and thoughts on a variety of different mediums. A very efficient way of doing this is using logograms, like Chinese have done for ages. Another method is to represent each syllable in a word by a symbol, like the Japanese do when writing telegrams. However, the most common way of storing characters is by using a limited set of shapes representing basic sounds (a.k.a. phonemes). Such a collection is called an *alphabet*, and the shapes are called *letters*.

TeX is primarily meant for typesetting languages that use this third method. The other two methods can also be dealt with, but some extra effort is needed. In this chapter we will focus on languages that use alphabets, the other methods will be explained in later chapters.

The shapes representing the characters that make up an alphabet are more or less standardized, and thereby can be recognized by readers even if their details differ. A collection of pictures matching character shapes is called a *font*, and the pictures in a font are called *glyphs*.


From left to right you see a Computer Modern font, a Helvetica lookalike, a Times Roman lookalike and the Antiqua Torunska font, all scaled to 60pt. As you can see, quite some variation is possible and when intermixed, the result is not always pleasing to look at. The term *fonts collection* refers to a set of fonts combined together in such a way that the overall appearance on a page looks good and reading is as comfortable as possible.

Even within a single font design there can be variations. In the example above there are a light, a bold, an italic, and a bold italic *alternative* of a single font. Such a set of fonts with the same basic design is known as a *font family*.

The distance between the individual glyphs in a word and the actual glyphs that are used depends on the combinations of these glyphs. In the top line of the next sample, the gap between the b and the o as well as the distance between the o and the x is slightly altered. This is called kerning. Further, the separate glyphs for the f and the i have been combined into a single one. This is called ligaturing.

box
box
file
file
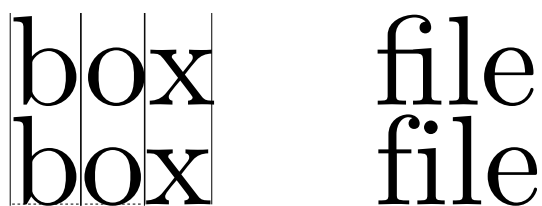
The font shown here is Computer Modern, the default TEX font. This font is designed by Donald Knuth. The Computer Modern has many kerning pairs, while the Palatino–like font that is used for most of the text in this manual has only a few, while both have essentially the same list of ligatures.

Micro–typography like kerning pairs and ligatures are not to be altered by the user, they are part of the font design and the required data is stored inside the font file, together with the drawing routines for the actual pictures. It *is* possible for the user to alter fonts and interline spacing and some more aspects on the level of macro–typography. The choice of font is the main topic of this chapter.

There are many different methods that can be used to classify fonts. There are classification systems based on the period in which the style was first developed; on the characteristics of the font; or the font application, like a newspaper or a book. Often, classification systems mix these characteristics up to some point.

For example, the Computer Modern family can be classified as a 'modern' font. This is a classification that primarily indicates a period (late 18th century), but it also implies a particular shape: 'modern' fonts have a high contrast between thick and thin strokes, and their stress axis is perfectly vertical.

At the same time, specific fonts in the Computer Modern family can be classified as 'serif' (glyphs strokes have embellishments at the end), 'sans serif' (shapes end abruptly), or 'monospaced' (all glyphs have the same width).

The Computer Modern family is in fact inspired by one font in particular: 'Modern 8a' by the Monotype corporation. Knuth implemented Computer Modern in METAFONT using parameters so that he could generate a whole collection of fonts all closely matching eachother in style. In CONTEXT you will normally use a reimplementation of Computer Modern using a more modern file format (Type 1 or OpenType). This new version is called 'Latin Modern', and also features an extended glyph set making it usable for languages that could not be typeset with Knuth's original fonts.

ok ok ok *OK* ok

In this example you see five font styles of Latin Modern: the Roman, Sans, Typewriter, Smallcaps and Variable Typewriter. Computer Modern is one of the few font families that comes with dedicated design sizes. The example below shows the differences of a 5, 7, 9, 12 and 17 point design scaled up to 48 points. Such nuances in font size are seldom seen these days.

As explained earlier, the general appearance of a font style can be classified according to many schemes. In table 1.1 you can see some examples of the naming of font styles that are often found together in a single document.

Serif	Sans	Mono
Regular	Support	TeleType
Roman	Sans	Type

Table 1.1 Some ways of classifying the styles in a font.

The top two series are normally used by typographers, the bottom series is what was traditionally used in plain TeX. In ConTeXt all three series of terms can be used because they are remapped to the same set of internal commands. As will be explained later, the command \rm is used to switch to a roman/serif/regular style, and \tt for switching to mono spaced or typewriter style, etcetera.

Text can be typeset in different font sizes. The unit `pt` is often used to specify the size. The availability of these font sizes are defined in definition files. Traditionally font designers used to design a glyph collection for each font size, but nowadays most fonts have a single design size of 10 points, or small set of sizes with names indicating their proposed use, like *caption*, *text*, and *display*.

The next sections will go into the details of switching of font styles and fonts in your documents. Be warned that the font switching mechanism is rather complex. This is caused by the different modes like math mode and text mode in ConTeXt. If you want to be understand the mechanism fully, you will have to acquaint yourself with the concept of encoding vectors and obtain some knowledge on fonts and their peculiarities.

1.2 The mechanism

Font switching is one of the eldest features of ConTeXt because font switching is indispensable in a macro package. During the years extensions to the font switching mechanism were inevitable. The following starting points have been chosesn during the development of this mechanism:

- To change a *style* must be easy, this means switching to: roman (serif, regular), sans serif (support), teletype (monospaced) etc. (\rm, \ss, \tt etc.)
- More than one *alternative* set of glyphs shapes must be available like slanted and bold (\sl and \bf).
- Different font *families* like Latin Modern Roman and Lucida Bright must be supported.
- It must be possbile to combine different families into font *collections*.

- Different sub– and superscripts must be available. These script sizes have to be retained across the switching of family, style and alternative.
- It should be possible to combine all of these requirements into a single definition unit called a *body font*.
- Changing the global font collection as well as the size must also be easy, and so sizes between 8pt and 14.4pt must be available by default.

Before reading further, please stop for a moment to make sure you thoroughly comprehend the above paragraphs. ConTeXt's terminology probably differs from what you are accustomed to, especially if you were previously a LaTeX user.

1.3 Font switching

The mechanism to switch from one style to another is somewhat complex, not in the least because the terminology is a bit fuzzy. A quick recap: we call a collection of fonts, like Lucida or Computer Modern Roman, a *family*. Within such a family, the members can be grouped according to characteristics. Such a group is called a *style*. Examples of styles within a family are: 'roman', 'sans serif' and 'teletype'. We saw already that there can be alternative classifications, but they all refer to the presence of serifs and the glyphs having equal widths. Within a style there can be *alternatives*, like 'boldface' and 'slanted'.

There are different ways to change into a new a style or alternative. You can use \ss to switch to a sans serif font style and \bf to get a bold alternative. When a different style is chosen, the alternatives adapt themselves to this style. Often a document will be typeset using just one combination of family and style. This is called the bodyfont.

Consistent use of commands like \bf and \sl in the text will automatically result in the desired bold and slanted altermatives when you change the family or style in the setup area of your input file.

1.3.1 Font style switching

Switching to another font style is done by one of five two-letter commands that are listed in table 1.2.

\rm	serif, regular, roman, rm
\ss	sans, support, sansserif, ss
\tt	mono, type, teletype, tt
\hw	handwritten, hw
\cg	calligraphic, cg
–	mm

Table 1.2   Font style switching commands

The 'handwritten' and 'calligraphic' font styles are sometimes useful when dealing with very elaborate document layout definitions. In the ConTeXt distribution, only the Lucida font family uses these styles, in any other font set they are simply ignored. You could use them in your own font setups if you so desire (see the next chapter for font setup definitions).

There is a sixth internal style that is only ever referenced as 'mm'. This style handles math fonts. It does not make sense to use this style directly so there is no command attached to it, but it is quite important internally so it makes sense to introduce it right away.

1.3.2 Font alternative switching

The alternatives within a style are given in table 1.3. Not all fonts have both italic and slanted or the bold alternatives of each. Some other fonts do not have small caps or only one set of digits. When an alternative is not known, ConTEXt will attempt to choose a suitable replacement automatically. For instance, the italic alternative may be used for if slanted is not available or vice versa.

\bf	bold
\sl	slanted
\it	italic
\bs	boldslanted, slantedbold
\bi	bolditalic, italicbold
\sc	smallcaps
\os	mediaeval (from *oldstyle*)
\tf	normal (from *typeface*)

Table 1.3   Font alternative switching commands and their keyword equivalents. With \os   you tell ConTEXt that you prefer mediaeval or old–style numbers as in 139 over 139.

Besides these two-letter commands, there is a series of font selector commands with a suffix attached. Some examples of that are:

```
\tfx \bfx \slx \itx
\tfa \tfb \tfc \tfd \tfxx
```

Each of the ordered alphabetic suffixes a, b, ... select a somewhat larger actual font than the previous one. The x and xx suffixes select smaller and yet smaller versions.

\bfx	smallbold
\slx	smallslanted
\itx	smallitalic
\bsx	smallboldslanted, smallslantedbold
\bix	smallbolditalic, smallitalicbold
\tfx	small, smallnormal

Table 1.4   Small alternative switching commands and their keyword equivalents.

Besides the 'small' switches that are mentioned in table 1.4, it depends on the completeness of the font definition files whether commands like \ita, \bfxx, \bfc, etc. are available. For the core ConTEXt fonts, you can count on at least \tfa, \tfb, \tfc, \tfd, and \tfxx being defined. For the others, just try and see what happens.

When you have chosen a larger charactersize, for example `\tfb`, then `\tf` equals `\tfb`, `\bf` equals `\bfb`, etc. This method is almost always preferable over returning to the original character size, but it may catch you off-guard.

More generic font scaling commands are also available:

```
\tx \txx
\setsmallbodyfont \setbigbodyfont
```

The command `\tx` adapts itself to both the style and the alternative. This command is rather handy when one wants to write macros that act like a chameleon. Going one more step smaller, is possible too: `\txx`. Using `\tx` when `\tx` is already given, is equivalent to `\txx`.

The commands `\setsmallbodyfont` and `\setbigbodyfont` switch to the 'small' and 'big' body font sizes. These relative sizes are defined via the 'body font environment', see section 1.8.

The various commands will adapt themselves to the actual setup of font and size. For example:

```
{\rm test {\sl test} {\bf test} \tfc test {\tx test} {\bf test}}
{\ss test {\sl test \tx test} {\bf test \tx test}}
```

will result in:

test *test* **test** test test **test**

test test test test test

When the `\rm` style is active, CONTEXT will interpret the command `\tfd` as if it was `\rmd`, when the style `\ss` is active, `\tfd` as is treated as `\ssd`. All default font setups use tf–setups so they will automatically adapt to the current font style.

The remainder of this paragraph is for completeness' sake only. Use of the following commands in new documents is discouraged.

Frequent font switching leads to longer processing times. When no sub- or superscripts are used and you are very certain what font you want to use, you can perform fast font switches with: `\rmsl`, `\ssbf`, `\tttf`, etc.

The plain TEX compatible font switches `\vi`, `\vii`, `\viii`, `\ix`, `\x`, and `\xii` are also defined, these have local effects like `\tfx` and `\tfa`.

1.3.3   Switching font styles in setup commands

A number of CONTEXT commands use the parameter `style` to set the used font. The parameter mechanism is rather flexible so that within the parameter `style` you can use any of the font switching commands like or bf or `\switchtobodyfont`, but also a number of keywords like

```
normal  bold  slanted  boldslanted  italic  bolditalic  type
small  smallbold  smallslanted  ...  smallitalic  ...  smalltype
capital
```

Most of these keywords have alread been listed in the tables 1.3 and 1.4, but a few predefined ones are still missing. These are displayed in table 1.5, together with the commands they execute. As is normal in CONTEXT, you can extend the list of accepted keywords by defining your own. This will be explained in section ?? in the next chapter.

`\tt`	type, mono
`\ttx`	smalltype
`\ss`	sans, sansserif
`\ss \bf`	sansbold
`\setsmallbodyfont`	smallbodyfont
`\setbigbodyfont`	bigbodyfont
`\smallcapped`	cap, capital
`\WORD`	WORD

Table 1.5 Remaining font alternative keywords.

1.4 Line spacing

In TeX linespacing is determined by a number of variable dimensions like `\topskip`, `\parskip` and `\baselineskip`. However, in ConTeXt these variables are related to the bodyfont size.

A line has a height and a depth. The distance between two lines is normally equal to the sum of the maximum height and maximum depth:

■ + ▬ = ■

This sum is in ConTeXt equal to 2.8ex, so almost three times the height of an x. This is about $1.2\times$ the bodyfont height. The proportion between maximum height and depth is default $.72 : .28$. Linespacing alters when a new bodyfont is used or when linespacing is defined explicitly by:

`\bfd \setupinterlinespace Now, the interline spacing is larger ... \par`

One has to consider the fact that TeX operates on paragraphs and within a group one has to close the paragraph explicitly with an empty line or `\par`. However, in most cases ConTeXt will take care of this.

Sometimes a line does not have the maximum height or depth. The next example illustrates this:

■■■ ■■■■■ ■■■ ■■■■■ ■■ ■■■■ ■■■■■.

It says:

The height and depth of lines differs.

When we put two of these lines above each other we will get:

■■■ ■■■■■ ■■■ ■■■■■ ■■ ■■■■ ■■■■■.
■■■ ■■■■■ ■■■ ■■■■■ ■■ ■■■■ ■■■■■.

You can see that the distance is somewhat bigger that the sum of the height and depth of each separate line. This distance is called the baseline distance (`\baselineskip`) and is in this document 13.8292pt. If we add some extra height to the line we see this:

■■■ ■■■■■ ■■■ ■■■■■ ■■ ■■■■ ■■■■■■
■■■ ■■■■■ ■■■ ■■■■■ ■■ ■■■■ ■■■■■.

To prevent the lines from touching TeX adds a `\lineskip`, in our example 1.0pt. In a similar way TeX is taking care of the first line of a page to have at least a height of `\topskip` (here 11.0pt plus 55.0pt).

Linespacing is set up by:

```
\setupinterlinespace [...]
                      *
                     OPTIONAL
 *   reset  small  medium  auto  big  on  off
```

Linespacing adapts to the size of the actual bodyfont automatically. This means that the user can leave this command untouched, unless a different linespacing is wanted. Instead of a factor one of the predetermined values `small` (1.0), `medium` (1.25) or `big` (1.5) can be given. Below an example is given of a text with a linespacing of 1.25: `\setupinterlinespace[medium]`.

> Whenever it comes to my mind that "everything that comes in quantities, will somehow survive", I also got the feeling that in a few hundred years people will draw the saddening conclusion that all those top–ten hits produced by computers represent the summum of todays musical and instrumental abilities. Isn't it true that archeologists can spend a lifetime on speculating about some old coins from the first century? On the other hand, the mere fact that one can have success with this type of non–music success of some top–hit musicians demonstrates both the listeners inability to rate the product and the lack of self criticism of the performers. In principle the future archeologist will therefore draw the right conclusion.

When you make a fontswitch the linespacing is adapted when you give the command `\setupinterlinespace` without any setup parameters and also when you add the key `reset`, for example

`\setupinterlinespace[reset,medium]`

The text below is typeset in the fontsize `\tfa`, using the following input:

```
\start \tfa \setupinterlinespace
In books meant for children we often find
a somewhat ... when needed. \par \stop
```

In this example the `\par` is necessary because otherwise TeX will have forgotten the linespacing before the paragraph is finished (in that case, the paragraph is ended by the empty line after the `\stop`).

> In books meant for children we often find a somewhat bigger typeface, for instance because we are convinced that this enables them to read the book themselves. On the other hand, I can also imagine that it is a cheap way to increase the number of pages. Unfortunately scaling up will also uncover the lack of quality of the typesetting used and/or the lack of typographic knowledge of the user of such a system. The interline space sometimes differs on a line by line basis, and depends on the height of the current line. Therefore, when changing the style, something that

should only be done on purpose, also change the baseline distance when needed.

Instead of a keyword, one can pass a key–value pair to define the characteristics of a line.

```
\setupinterlinespace [..,.=.,..]

*   height  =  NUMBER
    depth   =  NUMBER
    line    =  DIMENSION
    top     =  NUMBER
    bottom  =  NUMBER
```

The default settings are:

```
\setupinterlinespace
  [height=.72,
   depth=.28,
   top=1.0,
   bottom=0.4,
   line=2.8ex]
```

The `height` and `depth` determine the ratio between the height and depth of a line. The baseline distance is set to 2.8ex. The parameters `top` and `bottom` specify the relation between the bodyfont size and the height of the first line and the depth of the last line on a page. The last two quantities are related to TEX's `\topskip` and `\maxdepth`.

We will see later that instead of setting the spacing at the document level, i.e. for each font, you can set the spacing per body font environment:

```
\setupbodyfontenvironment
  [modern] [12pt]
  [interlinespace=14pt]
```

1.5   Emphasize

Within most macropackages the command `\em` is available. This command behaves like a chameleon which means that it will adapt to the actual typeface. In ConTEXt `\em` has the following characteristics:

- a switch to *slanted* or *italic* is possible
- a switch within `\bf` results in **bold slanted** or **bold italic** (when available)
- a so called *italic correction* is performed automatically (`\/`)

The bold italic or bold slanted characters are supported only when `\bs` and `\bi` are available.

```
The mnemonic {\em em} means {\em emphasis}.
{\em The mnemonic {\em em} means {\em emphasis}.}
{\bf The mnemonic {\em em} means {\em emphasis}.}
{\em \bf The mnemonic {\em em} {\em emphasis}.}
```

```
{\it The mnemonic em {\em means \bf emphasis}.}
{\sl The mnemonic em {\em means \bf emphasis}.}
```

This results in:

The mnemonic *em* means *emphasis*.
The mnemonic em *means* emphasis*.*
The mnemonic *em* means *emphasis*.
The mnemonic **em emphasis***.*
The mnemonic em means **emphasis***.*
The mnemonic em means **emphasis***.*

The advantage of the use of \em over \it and/or \sl is that consistent typesetting is enforced.

By default emphasis is set at *slanted*, but in this text it is set at *italic*. This setting is made via \setupbodyfontenvironment, see section 1.8 for more details:

```
\setupbodyfontenvironment
    [default]
    [em=italic]
```

1.6 Capitals

Words and abbreviations can be typeset in capitals. Both small and big characters are converted into capitals. When \cap is used to typeset a capital the size is that of an \tx. When you switch to slanted (\sl), bold (\bf), etc. the capital letter will also change. Since \cap has a specific meaning in math mode, the formal implementation is called \smallcapped. However in text mode one can use \cap.

```
\cap {..*..}

*   TEXT
```

```
\Cap {..*..}

*   TEXT
```

```
\CAP {..*..}

*   TEXT
```

```
\Caps {.. ..*.. ..}

*   WORD
```

The first command converts all letters to a capital. You are advised not to type capital letters in your source file because real small caps distinguishes between small and big letters.

```
Capitals for \cap {UK} are \cap {OK} and capitals for \cap {USA} are
okay. But what about capitals in \cap {Y2K}.
```

this results in:

Capitals for UK are OK and capitals for USA are okay. But what about capitals in Y2K.

A \cap within a \cap will not lead to any problems:

```
\cap {People that have gathered their \cap {capital} at the cost of other
people are not seldom \nocap {decapitated} in revolutionary times.}
```

or:

PEOPLE THAT HAVE GATHERED THEIR CAPITAL AT THE COST OF OTHER PEOPLE ARE NOT SELDOM decapitated IN REVOLUTIONARY TIMES.

In this example you can see that \cap can be temporarily revoked by \nocap.

```
\nocap {...}

*    TEXT
```

The command \Cap changes the first character of a word into a capital and \CAP changes letters that are preceded by \\ into capital letters. With \Caps you can change the first character of several words into a capital letter.

```
\setupcapitals [..,..=..,..]

*    title  = yes  no
     sc     = yes  no
```

With this command the capital mechanism can be set up. The key sc=yes switches to real SMALL CAPS. The key title determines whether capitals in titles are changed.

Next to the former \cap−commands there are also:

```
\Word {...}

*    WORD
```

and

```
\Words {.. ... ..}

*    WORD
```

These commands switch the first characters of a word or words into capitals. All characters in a word are changed with:

```
\WORD {.⃰.}

    *    WORD
```

Let's end this section with real small capitals. When these are available the real small caps \sc
are preferred over the pseudo–capital in abbreviations and logos.

```
In a manual on \TeX\ and Con\TeX t there is always the question whether
to type \cap{\TeX} and \cap{Con\TeX t} or {\sc \TeX} and {\sc Con\TeX t}.
Both are defined as a logo in the style definition so we type \type {\TEX}
and \type {\CONTEXT}, which come out as \TEX\ and \CONTEXT.
```

Results in:

In a manual on TEX and ConTEXt there is always the question whether to type TEX and CONTEXT
or TEX and ConTEXt. Both are defined as a logo in the style definition so we type \TEX and
\CONTEXT, which come out as TEX and ConTEXt.

IT IS ALWAYS POSSIBLE TO TYPESET TEXT IN SMALL CAPITALS. HOWEVER, REALIZE THAT LOWER CASE CHAR-
ACTERS DISCRIMINATE MORE AND MAKE FOR AN EASIER READ.

An important difference between \cap and \sc is that the last command is used for a specific
designed font type. The command \cap on the other hand adapts itself to the actual typeface:
KAP, **KAP**, *KAP*, etc.

Some typesetting packages stretch words (inter character spacing) to reach an acceptable align-
ment. In ConTEXt this not supported. On purpose! Words in titles can be stretched by:

```
\stretched {.⃰.}

    *    WORD
```

```
\hbox to \hsize {\stretched{there\\is\\much\\stretch\\in ...}}
\hbox to 20em   {\stretched{... and\\here\\somewhat\\less}}
```

With \\ you can enforce a space ({} is also allowed).

```
t  h  e  r  e      i  s      m  u  c  h      s  t  r  e  t  c  h      i  n      .  .  .
...  a n d  h e r e  s o m e w h a t  l e s s
```

These typographically non permitted actions are only allowed in heads. The macros that take
care of stretching do this by processing the text character by character.

This chapter will not go into the details of underlining because using underlining for typo-
graphical purposes is a bad practice. Instead, the commands related to under- and overlining
are discussed in section ?? ("??").

1.7   Selecting bodyfonts

The bodyfont (main font), font style and size is set up with:

```
\setupbodyfont [...,*...]

*    IDENTIFIER  serif  regular  roman  sans  support  sansserif  mono  type  teletype
     handwritten  calligraphic  5pt  ...  12pt
```

In a running text a temporary font switch is done with the command:

```
\switchtobodyfont [...,*...]

*  5pt  ...  12pt  small  big
```

This command doesn't change the bodyfont in headers and footers. With `small` and `big` you switch to a smaller or larger font.

In most cases, the command `\setupbodyfont` is only used once: in the style definition, and font switching inside the document is done with `\switchtobodyfont`. Don't confuse these two because that may lead to some rather strange but legitimate effects.

1.7.1 Body font sizes

Body font sizes actually consist of two components. Of course if you specify a size it directly specifies the size at which the main font is loaded, but a number of indirect parameters have to taken care of as well. Think of things like the font size used in headers, footers, footnotes, sub– and superscripts, as well as the interline space and a few others.

This is why in ConTEXt there is the concept of a *body font environment* (expressed as a dimension), and that is what you pass as an argument to `\setupbodyfont` or `\switchtobodyfont`. The definitions as presented above use the indication `5pt ... 12pt` for the body font environment, but actually any dimension is acceptable.

The most frequently used sizes are predefined as body font environments: 4pt . . . 12pt, 14.4pt, and 17.3pt. But when you use a different, not-yet-defined size specification —for example for a titlepage— ConTEXt will define a body font environment for that size automatically. While doing so, ConTEXt normally works with a precision of 1 decimal to prevent unnecessary loading of fontsizes with only small size differences.

Be warned that in this case, the results may be a less than ideal. The reason is that ConTEXt not just has to load the actual font, but it also has to guess at the various other settings like the relative font sizes and the interline space. It does so by using the values from the nearest smaller body font environment is that is already defined.

You can extend the list of predefined body font environments and even alter the precision in body font matching. See the section 1.8 for detailed information about how to tweak or define your own body font sizes.

To end this section, the example below demonstrates how the interline space is adapted automatically, when changing the size of the bodyfont. Consider this input:

```
{\switchtobodyfont[14.4pt] with these commands \par}
{\switchtobodyfont[12pt] for font switching  \par}
{\switchtobodyfont[10pt]  it is possible to   \par}
```

```
{\switchtobodyfont[8pt]  produce an eyetest: \par}
{\switchtobodyfont[6pt]  a x c e u i w m q p \par}
```

The actual ConTEXt behaviour is shown below on the left. On the right you can see what would have happened if the interline space were not automatically adapted.

with these commands
for font switching

it is possible to

produce an eyetest:
a x c e u i w m q p

with these commands
for font switching

it is possible to

produce an eyetest:

a x c e u i w m q p

1.7.2 Body font identifiers

In the definition block of `setupbodyfont` there was a list of words given besides the special marker `IDENTIFIER`. These words are the symbolic ConTEXt names for the font styles that we ran into earlier, with a few aliases so that you do not have to worry about the actual naming convention used. The symbolic names are mapped to two-letter internal style abbreviations that are used internally, see table 1.2 for an overview.

Although the macro syntax does not say so, you can use two-letter internal style abbreviations (`ss`, `rm`) as well as the longer names, if you prefer.

We have seen already that there are other and easier ways to switch the font style, so if `\setup-bodyfont` could only be used for this purpose it would not be all that useful. But luckily there is more: the optional `IDENTIFIER` can be a 'body font name' (aka 'typeface'). Such names have to be predefined, perhaps in a font support file, or simply on earlier lines in the style definition.

A 'typeface' is a symbolic name that links a single font style to actual font families. Such symbolic names are typically grouped together in a definition block that sets up values that link the four styles `\rm`, `\ss`, `\tt` and `\mm` to fonts in a 'font collection', and such definition blocks are called 'typescripts'.

ConTEXt expects you to define your own font setups, but there are quite a few examples predefined in various typescript files. Not all of those are perpetually loaded, so you usually have to execute a typescript explicitly to get the typeface names predefined. To this end, typescripts *themselves* also have names.

Executing a typescript is done by `\usetypescript`. We will get back to `\usetypescript` later because it is in fact a very flexible command, but let's discuss simple usage first.

```
\usetypescript [...¹...] [...²...]
                        OPTIONAL

1   IDENTIFIER

2   IDENTIFIER
```

A typical input sequence for selecting the predefined 'palatino' set of typefaces in MkII will look like this:

```
\usetypescript[palatino][ec]
\setupbodyfont[palatino,12pt]
```

In this example the typescript named `palatino` is asked for in the `ec` font encoding, and that defines a set of typefaces under the name `palatino`. These are then used by `\setupbody-font` and eventually this makes PDFTEX load the free Type 1 font URW Palladio in the correct encoding. URW Palladio is a font that looks a lot like the commercial font Linotype Palatino by Hermann Zapf, which explains the name of the typescript and typefaces.

Font encodings will be handled fully in the section 1.14. For now, please take for granted the fact that PDFTEX needs a second argument to `\usetypescript` that specifies an encoding name, and that there is a fixed set of acceptable names that depends on the typescript that is being requested.

In XƎTEX and MkIV the situation is a little bit different because fonts are reencoded to match Unicode whenever that is possible. That in turn means that XƎTEX and MkIV prefer to use OpenType fonts over Type 1 fonts, so different typescript definitions are used behind the scenes, and the second argument to `\usetypescript` becomes optional.

For example,

```
\usetypescript[palatino]
\setupbodyfont[palatino,12pt]
```

will make XƎTEX and LuaTEX load the OpenType font Pagella. This is a free font from the TEX Gyre project, that also looks just like the commercial font Linotype Palatino. You may as well leave the second argument in place: while it will always be ignored by LuaTEX, XƎTEX will actually use that encoding if the typescript uses Type 1 fonts instead of the more modern OpenType or TrueType font formats.

All predefined typescripts attach meaning to (at least) the three basic text font styles, so you can e.g. do this:

```
\usetypescript[times][ec]
\setupbodyfont[times,sans,12pt]
```

and end up using the OpenType font TEX Gyre Heros or the Type 1 font URW Nimbus Sans L. Both fonts are very similar in appearance to Linotype Helvetica, by the way.

The typescripts that come with the CONTEXT distribution are placed in source files that have names that start with `type-`. Some of these files are automatically loaded when needed, but most have to be loaded explicitly. There is a list in table 1.6

Some of the internal building blocks for typescripts are themselves located in yet other files (font size and font map file information, for example). Normally, when CONTEXT has to load typescript information from files, it will try to save memory by only executing the typescript it needs at that moment and discarding all other information. If you have enough memory at your disposal, you can speed up typescript use considerably by adding

```
\preloadtypescripts
```

in your preamble or your `cont-usr.tex`. This will make CONTEXT store all the typescript information in internal token registers the first (and therefore only) time it loads the actual files.

Explicit loading one of those files is done via the macro `\usetypescriptfile`.

The predefined typescripts, the typefaces they define, the files they are contained in inside the CONTEXT distribution, and the encodings they support in MkII mode are listed in table 1.7.

File	Loaded by PDFTEX	Loaded by XƎTEX	Loaded by MkIV	Description
type-akb	no	no	no	PostScript fonts using psnfss names (Type 1)
type-buy	no	no	no	Various commercial fonts (Type 1)
type-cbg	no	no	no	Greek free fonts (Type 1)
type-cow	no	no	no	The ConTEXt cow font (Type 1)
type-exp	no	no	no	Commercial Zapf fonts (OpenType)
type-fsf	no	no	no	Commercial Fontsite 500 fonts (Type 1)
type-ghz	no	no	no	Commercial Zapf fonts (Type 1)
type-gyr	no	no	no	The TEX Gyre project fonts (Type 1)
type-hgz	no	no	no	Commercial Zapf fonts (OpenType)
type-msw	no	no	no	Fonts that come with Microsoft Windows (Type 1)
type-omg	no	no	no	Omega free fonts (Type 1)
type-one	yes	no	no	Various free fonts (Type 1)
type-otf	no	yes	yes	Various free fonts (OpenType)
type-xtx	no	yes	no	Fonts that come with MacOSX (OpenType)

Table 1.6   The typescript source files that are part of ConTEXt.

In the following section there is a table (1.8) that explains for each typescript what font set it attaches to each of the font styles.

```
\usetypescriptfile [.⃰..]

*    IDENTIFIER
```

For example, the following

```
\usetypescriptfile[type-buy]
\usetypescript[lucida][texnansi]
\setupbodyfont[lucida,12pt]
```

will make PDFTEX use the Lucida Bright font family. Because this is a commercial font, this only works correctly if you have actually bought and installed the fonts. This uses the `texnansi` encoding because that is the preferred encoding of the actual fonts.

This is a good moment to explain a little trick: because the various `type-xxx` files define the building blocks for typescripts as well as the actual typescripts, it is sometimes possible to alter the effect of a typescript by loading an extra typescript file. For example,

```
\usetypescriptfile[type-gyr]
\usetypescript[palatino][ec]
\setupbodyfont[palatino,12pt]
```

will result in PDFTEX using the Type 1 font Pagella from the TEX Gyre project instead of the older and less complete URW Palladio, because the definition of the building blocks for the `palatino` typescript that is in the `type-gyr` file overwrites the preloaded definition from the `type-one` file.

Two of the files in the ConTEXt distribution exist precisely for this reason:

```
type-gyr.tex
```
    maps the typical PostScript font names for the free URW fonts to the TEX Gyre set;
```
type-akb.tex
```

Typescript	Typeface	File	Encodings
OmegaLGC	omlgc	type-omg	(unspecified)
antykwa-torunska	antykwa	type-one, type-otf	texnansi,ec,8r,t2a
cbgreek	cbgreek	type-cbg	(unspecified)
cbgreek-all	cbgreek-all	type-cbg	(unspecified)
cbgreek-medium	cbgreek-medium	type-cbg	(unspecified)
cow	cow	type-cow	default
fourier	fourier	type-one	ec
iwona	iwona	type-one, type-otf	texnansi,ec,8r,t2a
iwona-heavy	iwona-heavy	type-one, type-otf	texnansi,ec,8r,t2a
iwona-light	iwona-light	type-one, type-otf	texnansi,ec,8r,t2a
iwona-medium	iwona-medium	type-one, type-otf	texnansi,ec,8r,t2a
lucida	lucida	type-buy	texnansi,ec,8r
lucidabfm	lucida	type-buy	texnansi,ec,8r
lucidabfm	lucidabfm	type-buy	texnansi,ec,8r
lucidaboldmath	lucida	type-buy	texnansi,ec,8r
lucidaboldmath	lucidaboldmath	type-buy	texnansi,ec,8r
modern	modern	type-one, type-otf	texnansi,ec,qx,t5,default
modern-base	modern	type-one, type-otf	texnansi,ec,qx,t5,default,t2a/b/c
modernvariable	modernvariable	type-one, type-otf	texnansi,ec,qx,8r,t5
optima	optima	type-one	texnansi,ec,qx
optima	optima	type-ghz	texnansi,ec,qx
optima-nova	optima	type-ghz, type-hgz	texnansi,ec
optima-nova-os	optima-os	type-ghz, type-hgz	texnansi,ec
palatino	palatino	type-hgz	(cannot be used in MκII)
palatino	palatino	type-one, type-otf	texnansi,ec,qx,8r,t5
palatino-informal	palatino-informal	type-hgz	(cannot be used in MκII)
palatino-light	palatino-light	type-exp	(cannot be used in MκII)
palatino-medium	palatino-medium	type-exp	(cannot be used in MκII)
palatino-normal	palatino-normal	type-exp	(cannot be used in MκII)
palatino-nova	palatino	type-hgz	(cannot be used in MκII)
palatino-sans	palatino	type-hgz	(cannot be used in MκII)
postscript	postscript	type-one, type-otf	texnansi,ec,qx,8r,t5
sheep	sheep	type-cow	default
times	times	type-one, type-otf	texnansi,ec,qx,8r,t5

Table 1.7 The typescripts. Typescripts that use commercial fonts are typeset in bold.

maps the same names to the commercial Adobe fonts.

For the defintions in the second file to work, you also need to execute an extra typescript:

```
\usetypescriptfile [type-akb]
\usetypescript [adobekb] [ec]

\usetypescript [palatino] [ec]
\setupbodyfont[palatino,12pt]
```

1.7.3   Typeface definitions

Defining a typeface goes like this:

```
\starttypescript [palatino] [texnansi,ec,qx,t5,default]

\definetypeface[palatino] [rm] [serif][palatino] [default]
```

```
\definetypeface[palatino] [ss] [sans] [modern]    [default] [rscale=1.075]
\definetypeface[palatino] [tt] [mono] [modern]    [default] [rscale=1.075]
\definetypeface[palatino] [mm] [math] [palatino] [default]
```

`\stoptypescript`

This defines a typescript named `palatino` in five different encodings. When this typescript is executed via `\usetypescript`, it will define four typefaces, one of each of the four basic styles `rm`, `ss`, `tt`, and `mm`.

```
\definetypeface [..1..] [..2..] [..3..] [..4..] [..5..] [..6..]
                                                    OPTIONAL OPTIONAL

1    TEXT
2    rm  ss  tt  mm  hw  cg

3    IDENTIFIER

4    IDENTIFIER

5    IDENTIFIER

6    features  =  IDENTIFIER
     rscale    =  NUMBER
     encoding  =  IDENTIFIER
     text      =  IDENTIFIER
```

The third and fourth arguments to `\definetypeface` are pointers to already declared font sets, these are defined elsewhere. Table 1.8 gives the full list of predefined typescripts (the first argument of `\starttypescript`) and font sets that are attached to the styles (the third and fourth argument of each `\definetypeface`).

The names in the third argument (like `serif` and `sans`) do *not* have the same meaning as the names used in `\setupbodyfont`. Inside `\setupbodyfont`, they were keywords that were internally remapped to one of the two-letter internal styles. Inside `\definetypeface`, they are nothing more than convenience names that are attached to a group of fonts by the person that wrote the font definition. They only reflect a grouping that the person believed that could be a single font style. Oftentimes, these names are identical to the official style keywords, just as the typescript and typeface names are often the same, but there can be (and are) different names altogether.

How to define your own font sets will be explained in the next chapter, but there are quite a few predefined font sets that come with ConTEXt; these are all listed in the four tables 1.9, 1.10, 1.11, and 1.12.

For everything to work properly in MkII, the predefined font sets also have to have an encoding attached, you can look those up in the relevant tables as well.

The fifth argument to `\definetypeface` specifies specific font size setups (if any), these will be covered in section ?? in the next chapter. Almost always, specifying `default` will suffice.

The optional sixth argument is used for tweaking font settings like the specification of font features or adjusting parameters. In this case, the two `modern` font sets are loaded with a small magnification, this evens out the visual heights of the font styles.

Typescript	Style rm	Style ss	Style tt	Style mm
OmegaLGC	omega	–	omega	–
antykwa-torunska	antykwa-torunska	modern	modern	antykwa-torunska
cbgreek	cbgreek	cbgreek	cbgreek	–
cbgreek-all	cbgreek	cbgreek	cbgreek	–
cbgreek-medium	cbgreek	cbgreek	cbgreek	–
cow	cow	cow serif	modern	cow
fallback	modern	modern	modern	modern
fourier	fourier	modern	modern	fourier
iwona	modern	iwona	modern	iwona
iwona-heavy	modern	iwona-heavy	modern	iwona-heavy
iwona-light	modern	iwona-light	modern	iwona-light
iwona-medium	modern	iwona-medium	modern	iwona-medium
lucida	lucida	lucida	lucida	lucida
lucidabfm	lucida	lucida	lucida	lucida bfmath
lucidaboldmath	lucida	lucida	lucida	lucida boldmath
modern	modern	modern	modern	modern
modern-base	(computer-)modern	(computer-)modern	(computer-)modern	(computer-)modern
modernvariable	simple	modern	modern	modern
optima	palatino	optima-nova	modern	palatino
optima-nova	optima-nova sans	optima-nova	latin-modern	latin-modern
optima-nova-os	optima-nova-os sans	optima-nova-os	latin-modern	latin-modern
palatino	palatino-nova	palatino-sans	latin-modern	latin-modern
palatino	palatino	modern	modern	palatino
palatino-informal	palatino-nova	palatino-informal	latin-modern	latin-modern
palatino-light	palatino-nova	palatino-sans-light	latin-modern	latin-modern
palatino-medium	palatino-nova	palatino-sans-medium	latin-modern	latin-modern
palatino-normal	palatino-nova	palatino-sans-normal	latin-modern	latin-modern
palatino-nova	palatino-nova	palatino-sans	latin-modern	latin-modern
palatino-sans	palatino-nova	palatino-sans	latin-modern	latin-modern
postscript	times	helvetica	courier	times
sheep	sheep	sheep serif	modern	sheep
times	times	helvetica	modern	times

Table 1.8 The typescripts.
Unless stated otherwise, style **rm** uses a group named serif, style **ss** uses sans,
style **tt** uses mono, and style **mm** uses math. A single dash in a cell means that
the typescript does not define that style, you should refrain from using the style.
The lucida, lucidabfm, and lucidaboldmath typescripts also define **hw** and **cg** as
'lucida handwring' and 'lucida calligraphy'. The modern-base typescript switch-
es back to computer-modern for a few legacy encodings: t2a, t2b, and t2c.

A note for the lazy: if the sixth argument is not given and the fifth argument happens to be
`default`, then the fifth argument can be omitted as well.

There are four possible keys in the sixth argument:

key	default value	explanation
rscale	1	a scaling factor for this typescript relative to the selected body font size
encoding	\defaultencoding	the encoding for the typeface, normally inherited from the typescript automatically
features		this applies a predefined font feature set (see section 1.9)
text		sets up the forced math text style

Identifier	file	Encodings	Supported styles
modern	type-one	ec, qx, texnansi, t5	serif, sans, mono, math, boldmath, bfmath
latin-modern	type-one	ec, qx, texnansi, t5	serif, sans, mono, math, boldmath, bfmath
computer-modern	type-one	t2a/b/c	serif, sans, mono, math, boldmath, bfmath
simple	type-one	– synonyms only –	serif
concrete	type-one	– hardcoded –	serif
euler	type-one	– hardcoded –	math, boldmath, bfmath
ams	type-one	– hardcoded –	math
fourier	type-one	ec	math, serif
courier	type-one	8r, ec, qx, texnansi, t5	mono
helvetica	type-one	8r, ec, qx, texnansi, t5	sans
times	type-one	8r, ec, qx, texnansi, t5	serif, math
palatino	type-one	8r, ec, qx, texnansi, t5	serif, math
bookman	type-one	8r, ec, qx, texnansi, t5	serif
schoolbook	type-one	8r, ec, texnansi, t5	serif
chancery	type-one	8r, ec, qx, texnansi	calligraphy
charter	type-one	8r, ec, texnansi	serif
utopia	type-one	ec, texnansi	serif
antykwa-torunska	type-one	ec, qx, texnansi, t5, t2a/b/c, greek	serif, math
antykwa-torunska-light	type-one	ec, qx, texnansi, t5, t2a/b/c, greek	serif, math
antykwa-torunska-cond	type-one	ec, qx, texnansi, t5, t2a/b/c, greek	serif, math
antykwa-torunska-lightcond	type-one	ec, qx, texnansi, t5, t2a/b/c, greek	serif, math
antykwa-poltawskiego	type-one	8r, ec, texnansi	serif
iwona	type-one	ec, qx, texnansi, t5	sans, math
iwona-light	type-one	ec, qx, texnansi, t5	sans, math
iwona-medium	type-one	ec, qx, texnansi, t5	sans, math
iwona-heavy	type-one	ec, qx, texnansi, t5	sans, math
iwona-cond	type-one	ec, qx, texnansi, t5	sans
iwona-light-cond	type-one	ec, qx, texnansi, t5	sans
iwona-medium-cond	type-one	ec, qx, texnansi, t5	sans
iwona-heavy-cond	type-one	ec, qx, texnansi, t5	sans
kurier	type-one	ec, qx, texnansi, t5	sans, math
kurier-light	type-one	ec, qx, texnansi, t5	sans, math
kurier-medium	type-one	ec, qx, texnansi, t5	sans, math
pagella	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
palatino	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
termes	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
times	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
bonum	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
bookman	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
schola	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
schoolbook	type-gyr	ec, qx, texnansi, t5, t2a/b/c	serif
heros	type-gyr	ec, qx, texnansi, t5, t2a/b/c	sans
helvetica	type-gyr	ec, qx, texnansi, t5, t2a/b/c	sans
adventor	type-gyr	ec, qx, texnansi, t5, t2a/b/c	sans
cursor	type-gyr	ec, qx, texnansi, t5, t2a/b/c	mono
courier	type-gyr	ec, qx, texnansi, t5, t2a/b/c	mono
omega	type-omg	– hardcoded –	naskh, serif, mono
cbgreek	type-cbg	– hardcoded –	serif, sans, mono
cbgreek-medium	type-cbg	– hardcoded –	serif, sans, mono
cbgreek-all	type-cbg	– hardcoded –	serif, sans, mono
cow	type-cow	– hardcoded –	math, serif
sheep	type-cow	– hardcoded –	math, serif

Table 1.9   The predefined body font identifiers for free Type 1 and METAFONT fonts

Identifier	file	Encodings	Supported styles
lucida	type-buy	8r, ec, texnansi	serif, sans, mono, handwriting, calligraphy, math, boldmath, bfmath, casual, fax
informal	type-buy	– hardcoded –	casual, math
officina	type-buy	8r, ec, texnansi	serif, sans
meta	type-buy	8r, ec, texnansi	serif, sans, expert
meta-medium	type-buy	8r, ec, texnansi	sans
meta-lf	type-buy	8r, ec, texnansi	sans
meta-book	type-buy	8r, ec, texnansi	sans
meta-book-lf	type-buy	8r, ec, texnansi	sans
meta-bold	type-buy	8r, ec, texnansi	sans
meta-bold-lf	type-buy	8r, ec, texnansi	sans
meta-normal	type-buy	8r, ec, texnansi	sans
meta-normal-lf	type-buy	8r, ec, texnansi	sans
meta-medium	type-buy	8r, ec, texnansi	sans
meta-medium-lf	type-buy	8r, ec, texnansi	sans
meta-black	type-buy	8r, ec, texnansi	sans
meta-black-lf	type-buy	8r, ec, texnansi	sans
univers	type-buy	8r, ec, texnansi	sans
univers-light	type-buy	8r, ec, texnansi	sans
univers-black	type-buy	8r, ec, texnansi	sans
mendoza	type-buy	8r, ec, texnansi	serif
frutiger	type-buy	8r, ec, texnansi	sans
kabel	type-buy	8r, ec, texnansi	sans
thesans	type-buy	8r, ec, texnansi	sans, mono, expert
sabon	type-buy	8r, ec, texnansi	serif
stone	type-buy	ec, texnansi	serif, sans
stone-oldstyle	type-buy	– synonyms only –	serif, sans
industria	type-buy	ec, texnansi	sans
bauhaus	type-buy	ec, texnansi	sans
swift	type-buy	ec, texnansi	serif
swift-light	type-buy	– synonyms only –	serif
syntax	type-buy	ec, texnansi	sans
linoletter	type-buy	ec, texnansi	serif
zapfino	type-ghz	8r, ec, texnansi	serif, handwriting
palatino-sans-light	type-exp	texnansi, ec	sans
palatino-sans-normal	type-exp	texnansi, ec	sans
palatino-sans-medium	type-exp	texnansi, ec	sans
opus	type-fsf	8r, ec, texnansi	sans
typewriter	type-fsf	8r, ec, texnansi	mono
garamond	type-fsf	8r, ec, texnansi	serif
optima	type-ghz	8r, ec, texnansi	sans
optima-nova	type-ghz	8r, ec, texnansi	sans
optima-nova-os	type-ghz	8r, ec, texnansi	sans
optima-nova-light	type-ghz	8r, ec, texnansi	sans
optima-nova-medium	type-ghz	8r, ec, texnansi	sans
palatino	type-ghz	8r, ec, texnansi	serif
palatino-nova	type-ghz	8r, ec, texnansi	serif
palatino-nova-os	type-ghz	8r, ec, texnansi	serif
palatino-nova-light	type-ghz	8r, ec, texnansi	serif
palatino-nova-medium	type-ghz	8r, ec, texnansi	serif
aldus-nova	type-ghz	8r, ec, texnansi	serif
melior	type-ghz	8r, ec, texnansi	serif
verdana	type-msw	texnansi	sans
arial	type-msw	texnansi	sans

Table 1.10   The predefined body font identifiers for commercial Type 1 fonts

Identifier	file	Supported styles	Identifier	file	Supported styles
modern	type-otf	serif, sans, mono, math, boldmath, bfmath	palatino	type-otf	serif, math
			times	type-otf	serif, math
			bookman	type-otf	serif
latin-modern	type-otf	serif, sans, mono, math, boldmath, bfmath	schoolbook	type-otf	serif
			chancery	type-otf	calligraphy
			helvetica	type-otf	sans
modern-vari	type-otf	mono	courier	type-otf	mono
latin-modern-vari	type-otf	mono	antykwa-torunska	type-otf	serif, math
modern-cond	type-otf	mono	antykwa-torunska-light	type-otf	serif, math
latin-modern-cond	type-otf	mono	antykwa-torunska-cond	type-otf	serif, math
computer-modern	type-otf	serif, sans, mono, math, boldmath, bfmath	antykwa-torunska-lightcond	type-otf	serif, math
			antykwa-poltawskiego	type-otf	serif
			iwona-light	type-otf	sans, math
concrete	type-otf	serif	iwona	type-otf	sans, math
euler	type-otf	math, boldmath, bfmath	iwona-medium	type-otf	sans, math
			iwona-heavy	type-otf	sans, math
ams	type-otf	math	iwona-cond	type-otf	sans
pagella	type-otf	serif	iwona-light-cond	type-otf	sans
termes	type-otf	serif	iwona-medium-cond	type-otf	sans
bonum	type-otf	serif	iwona-heavy-cond	type-otf	sans
schola	type-otf	serif	kurier	type-otf	sans, math
chorus	type-otf	serif	kurier-light	type-otf	sans, math
heros	type-otf	sans	kurier-medium	type-otf	sans, math
adventor	type-otf	sans	charter	type-otf	serif
cursor	type-otf	sans	gentium	type-xtx	serif

Table 1.11   The predefined body font identifiers for free Unicode (Opentype) fonts

Identifier	file	Supported styles	Identifier	file	Supported styles
zapfino	type-hgz	serif, handwriting	times	type-xtx	serif
optima-nova	type-hgz	sans	palatino	type-xtx	serif
optima-nova-os	type-hgz	sans	helvetica	type-xtx	sans
optima-nova-light	type-hgz	sans	courier	type-xtx	mono
optima-nova-medium	type-hgz	sans	hoefler	type-xtx	serif
palatino-nova	type-hgz	serif	lucidagrande	type-xtx	sans
palatino-nova-os	type-hgz	serif	optima	type-xtx	sans
palatino-nova-light	type-hgz	serif	gillsans	type-xtx	sans
palatino-nova-medium	type-hgz	serif	gillsanslt	type-xtx	sans
palatino-sans	type-hgz	sans	zapfino	type-xtx	handwriting, serif
palatino-informal	type-hgz	sans	applechancery	type-xtx	calligraphy, serif
melior	type-hgz	serif	timesnewroman	type-xtx	serif
– all four-variant fonts –	type-xtx	Xserif	arial	type-xtx	sans
– all four-variant fonts –	type-xtx	Xsans	lucida	type-xtx	serif, sans, mono, handwriting, fax, calligraphy
– all four-variant fonts –	type-xtx	Xmono			

Table 1.12   The predefined body font identifiers for commercial Unicode (Opentype) fonts

If you look closely, in table 1.12 you will notice three very special items: `Xserif`, `Xsans` and `Xmono`. These belong to a special X∃TEX-only trick called 'wildcard typescripts'.

X∃TEX offers some nice features in terms of automatically finding related fonts in a family, namely the italic, bold, and bolditalic alternatives. To take advantage of that, there's a set of wildcard

typescripts that take an arbitrary Macintosh font name as input, and provide as many of the alternatives it can find. To set these typescripts (and the calling conventions) apart from the familiar ones, the typescripts are identified with Xserif, Xsans, and Xmono.

To call the typescripts, it's most convenient to define a typeface that uses these features. The named font slot should contain the display name of the Regular alternative (not the family name) of the font in question. For example, you could have the following mix:

```
\starttypescript[myface]
\definetypeface[myface][rm][Xserif][Baskerville]   [default]
\definetypeface[myface][tt][Xmono] [Courier]       [default][rscale=.87]
\definetypeface[myface][ss][Xsans] [Optima Regular][default]
\stoptypescript
```

As you can see, you can activate relative scaling of face sizes. The above definitions look very much like any other typeface definition, except that the serif/sans/mono identifier is preceded with X, and that there is no underlying "Optima Regular" defined anywhere. Those missing bits of the definitions are handled by typescript and X∃TEX magic.

1.8 Body font environments

Earlier we have seen that within a font family there are different font sizes. The relations between these sizes are defined by *body font environments*.

For all regular font sizes, environments are predefined that fulfill their purpose adequately. However when you want to do some extra defining yourself there is:

```
\definebodyfontenvironment [..1..] [..2..] [..,..3..,..]
                              OPTIONAL         OPTIONAL

1    IDENTIFIER

2    5pt   ... 12pt  default

3    text            = DIMENSION
     script          = DIMENSION
     scriptscript    = DIMENSION
     x               = DIMENSION
     xx              = DIMENSION
     small           = DIMENSION
     big             = DIMENSION
     interlinespace  = DIMENSION
     em              = normal  bold  slanted  boldslanted  type  cap  small...  COMMAND
```

The first argument is optional, and specifier the typeface identifier that this particular body font environment setup is for. It defaults to the current typeface.

The second argument is the size of the body font environment that is being defined. This argument is not really optional, the macro syntax description is a little misleading.

The third argument once again is optional, and contains the actual settings as key-value pairs. If it is missing, defaults will be guessed at by CONTEXT itself.

text	Math text size
script	Math script size
scriptscript	Math scriptscript size
x	The size used for the \tfx etc. commands
xx	The size used for the \tfxx command
big	The 'larger' font size
small	The 'smaller' font size
interlinespace	Distance between lines in a paragraph, can be ignored because a reasonable default is always set up
em	The style to use for emphasis (default is slanted)

So, when you want to have a somewhat bigger fontsize for just a few words (e.g. for a book title) you can type:

```
\definebodyfontenvironment [24pt]
\switchtobodyfont[24pt]
```

but longer stretches of text, you will need to set up most of the values, using something like this

```
\definebodyfontenvironment
  [22pt]
  [         text=22pt,
         script=17.3pt,
   scriptscript=14.4pt,
              x=17.3pt,
             xx=14.4pt,
            big=28pt,
          small=17.3pt]
```

To tweak already defined sizes, there is an accompagnying setup command with the same parameter conventions:

```
\setupbodyfontenvironment [...1..] [...2..] [..,..3.,..]
                              OPTIONAL         OPTIONAL

1    inherits from \definebodyfontenvironment

2   inherits from \definebodyfontenvironment

3   inherits from \definebodyfontenvironment
```

1.9   Font feature sets

As mentioned already, some fonts contain extra information besides the actual glyph shapes. In traditional TEX fonts, the extra information is roughly limited to kerning pairs and ligature information, and both of these 'features' are automatically applied to the text that is being typeset. In the odd case where one of the two needs to be suppressed, a little bit of macro trickery can do the job without too many complicating factors.

But with the new OpenType font format that is used by X$\overline{\text{H}}$TEX and LuaTEX, the list of possible features is increased enormously. OpenType fonts have not just kerning information and ligature information, but there can also be other features like optional oldstyle figures, caps and smallcaps glyphs, decorative swashes, etc. all inside a single font file.

Not only that, but some of these features are not even supposed to be active all the time. Certain features should only be activated if the user asks for it, other features depend on the script and language that is in use for the text that is being typeset.

This is a big step forward in that there are now far fewer fonts needed to achieve the same level of quality than before, all that extra font information also poses a big challenge for macro writers. And add to that the fact that at the core, the two engines (X$\overline{\text{H}}$TEX and LuaTEX) handle OpenType fonts completely different from each other.

CONTEXT has a new subsystem called 'font features' to create order in this forest of features. The most important command is \definefontfeature. This command can be used to group various font features under a single symbolic name, that can then be used as e.g. the argument to the features key of \definetypeface.

```
\definefontfeature [..¹..] [..²..] [..³..]
                                    OPTIONAL

1    TEXT
2    IDENTIFIER

3    compose    =  no  yes
     mode       =  node  base
     tlig       =  no  yes
     trep       =  no  yes
     script     =  IDENTIFIER
     language   =  IDENTIFIER
     ..tag..    =   no  yes
```

```
\definefontfeature
   [default-base]
   [script=latn,language=dflt,liga=yes,kern=yes,tlig=yes,trep=yes]
```

As you can probably guess, the first argument is the symbolic name that is being defined. The second argument is a mix of a-hoc settings and OpenType font features.

compose	Use fallback composition in MkIV (experimental, undocumented)
protrusion	Character protrusion in MkIV (see section 1.13)
expansion	Character expansion in MkIV (see section 1.13)
script	An OpenType script identifier
language	An OpenType script language identifier
tlig	A virtual feature for legacy (TEX-style) automatic ligatures (for compatibility, there is an alias for this key called texligatures)
trep	A virtual feature for legacy (TEX-style) automatic ligatures (for compatibility, there is an alias for this key called texquotes) (only works in MkIV)

mode   Processing mode for MkIV. `node` and `base` allowed, `base` is default
<tag>   Any OpenType feature tag is acceptable, but in MkIV only a 'known' subset ac-
                tually has any effect, and then only in `node` mode. This list is given in table 1.13.
                In X∃TEX, processing depends on the internal subengine that is used by X∃TEX,
                and that is outside of ConTEXt's control.

A few fontfeatures are predefined by context:

```
default     liga=yes,kern=yes,tlig=yes,trep=yes
smallcaps   liga=yes,kern=yes,tlig=yes,trep=yes,smcp=yes
oldstyle    liga=yes,kern=yes,tlig=yes,trep=yes,onum=yes
```

At the moment, `smallcaps` and `oldstyle` only work in X∃TEX (in MkIV, it would need an extra
`mode=node` pair).

1.10   Displaying the current font setup

With the command `\showbodyfont` an overview is generated of the available characters, and
an overview of the different fontsizes within a family can be summoned with `\showbody-`
`fontenvironment`.

```
\showbodyfont [...,*...]
                    OPTIONAL
*   inherits from \setupbodyfont
```

```
\showbodyfontenvironment [...,*...]
                            OPTIONAL
*   inherits from \setupbodyfont
```

Specifiying actual IDENTIFIERs to these commands is currently unreliable because they inter-
nally are still counting on an older system of body font definitions, but you can safely use a size
argument to get the information for the current font set.

Below an example of the possible output is shown, for `\showbodyfont[12pt]`

[palatino] [12pt]												\mr : Ag
	\tf	\sc	\sl	\it	\bf	\bs	\bi	\tfx	\tfxx	\tfa	\tfb	\tfc	\tfd
\rm	Ag	AG	*Ag*	*Ag*	**Ag**	*Ag*	*Ag*	Ag	Ag	Ag	Ag	Ag	Ag
\ss	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag	Ag
\tt	Ag	AG	*Ag*	*Ag*	**Ag**	*Ag*	*Ag*	Ag	Ag	Ag	Ag	Ag	Ag

And the output of `\showbodyfontenvironment[12pt]` is:

aalt	Access All Alternates	ital	Italics	smcp	Small Capitals
abvf	Above-Base Forms	jalt	Justification Alternatives	smpl	Simplified Forms
abvm	Above-Base Mark Positioning	jp04	JIS2004 Forms	ss01	Stylistic Set 1
		jp78	JIS78 Forms	ss02	Stylistic Set 2
abvs	Above-Base Substitutions	jp83	JIS83 Forms	ss03	Stylistic Set 3
afrc	Alternative Fractions	jp90	JIS90 Forms	ss04	Stylistic Set 4
akhn	Akhands	kern	Kerning	ss05	Stylistic Set 5
blwf	Below-Base Forms	lfbd	Left Bounds	ss06	Stylistic Set 6
blwm	Below-Base Mark Positioning	liga	Standard Ligatures	ss07	Stylistic Set 7
		ljmo	Leading Jamo Forms	ss08	Stylistic Set 8
blws	Below-Base Substitutions	lnum	Lining Figures	ss09	Stylistic Set 9
c2pc	Petite Capitals From Capitals	locl	Localized Forms	ss10	Stylistic Set 10
		mark	Mark Positioning	ss11	Stylistic Set 11
c2sc	Small Capitals From Capitals	medi	Medial Forms	ss12	Stylistic Set 12
		med2	Medial Forms #2	ss13	Stylistic Set 13
calt	Contextual Alternates	mgrk	Mathematical Greek	ss14	Stylistic Set 14
case	Case-Sensitive Forms	mkmk	Mark to Mark Positioning	ss15	Stylistic Set 15
ccmp	Glyph Composition/Decomposition	mset	Mark Positioning via Substitution	ss16	Stylistic Set 16
				ss17	Stylistic Set 17
cjct	Conjunct Forms	nalt	Alternate Annotation Forms	ss18	Stylistic Set 18
clig	Contextual Ligatures	nlck	NLC Kanji Forms	ss19	Stylistic Set 19
cpsp	Capital Spacing	nukt	Nukta Forms	ss20	Stylistic Set 20
cswh	Contextual Swash	numr	Numerators	subs	Subscript
curs	Cursive Positioning	onum	Old Style Figures	sups	Superscript
dflt	Default Processing	opbd	Optical Bounds	swsh	Swash
dist	Distances	ordn	Ordinals	titl	Titling
dlig	Discretionary Ligatures	ornm	Ornaments	tjmo	Trailing Jamo Forms
dnom	Denominators	palt	Proportional Alternate Width	tnam	Traditional Name Forms
expt	Expert Forms	pcap	Petite Capitals	tnum	Tabular Figures
falt	Final glyph Alternates	pnum	Proportional Figures	trad	Traditional Forms
fina	Terminal Forms	pref	Pre-base Forms	twid	Third Widths
fin2	Terminal Forms #2	pres	Pre-base Substitutions	unic	Unicase
fin3	Terminal Forms #3	pstf	Post-base Forms	valt	Alternate Vertical Metrics
frac	Fractions	psts	Post-base Substitutions	vatu	Vattu Variants
fwid	Full Width	pwid	Proportional Widths	vert	Vertical Writing
half	Half Forms	qwid	Quarter Widths	vhal	Alternate Vertical Half Metrics
haln	Halant Forms	rand	Randomize		
halt	Alternate Half Width	rkrf	Rakar Forms	vjmo	Vowel Jamo Forms
hist	Historical Forms	rlig	Required Ligatures	vkna	Vertical Kana Alternates
hkna	Horizontal Kana Alternates	rphf	Reph Form	vkrn	Vertical Kerning
hlig	Historical Ligatures	rtbd	Right Bounds	vpal	Proportional Alternate Vertical Metrics
hngl	Hangul	rtla	Right-To-Left Alternates		
hojo	Hojo Kanji Forms	ruby	Ruby Notation Forms	vrt2	Vertical Rotation
hwid	Half Width	salt	Stylistic Alternates	zero	Slashed Zero
init	Initial Forms	sinf	Scientific Inferiors		
isol	Isolated Forms	size	Optical Size		

Table 1.13 The OpenType features that are understood by MꜱIV in `mode=node` processing mode

[palatino] [12pt]							
text	script	scriptscript	x	xx	small	big	interlinespace
20.7pt	14.4pt	12pt	17.3pt	14.4pt	17.3pt	20.7pt	not set
17.3pt	12.1pt	8.6pt	13.8pt	10.3pt	13.8pt	20.7pt	not set
14.4pt	10pt	7.2pt	11.5pt	8.6pt	11.5pt	17.2pt	not set
12pt	8.3pt	6pt	9.6pt	7.2pt	9.6pt	14.3pt	not set
11pt	7.6pt	5.5pt	8.8pt	6.6pt	8.8pt	13.1pt	not set
10pt	6.9pt	5pt	8pt	6pt	8pt	11.9pt	not set
9pt	6.2pt	4.5pt	7.2pt	5.4pt	7.2pt	10.7pt	not set
8pt	5.5pt	4pt	6.4pt	4.8pt	6.4pt	9.5pt	not set
7pt	4.8pt	3.5pt	5.6pt	4.2pt	5.6pt	8.3pt	not set
6pt	4.1pt	3pt	4.8pt	3.6pt	4.8pt	7.1pt	not set
5pt	3.4pt	2.5pt	4pt	3pt	4pt	5.9pt	not set
4pt	2.7pt	2pt	3.2pt	2.4pt	3.2pt	4.7pt	not set

1.11   Math fonts

There are only a few font families in existance that can handle math properly because such fonts have to carry a complete set of characters and symbols for mathematical typesetting. Among these, the Computer Modern Roman distinguishes itself by its many design sizes; that really pays off when typesetting complicated math formulas.

Many TEX users have chosen TEX for its superb math typesetting.

This chapter will not go into any details but in math mode, the central concept is the *math family* (not to be confused with the *font families* discussed earlier). There are math families for \bf, \it, etc. as well as for the special math symbols. Within each family, there are always exactly three member fonts: `text`, `script` and `scriptscript`, or a normal, smaller and smallest font. The normal font size is used for running text and the smaller ones for sub and superscripts. The next example will show what the members of a math family can do.

```
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\rm 6x^2$
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\tf 6x^2$
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\bf 6x^2$
$\tf x^2+\bf x^2+\sl x^2+\it x^2+\bs x^2+ \bi x^2 =\sl 6x^2$
```

When this is typeset you see this:

$x^2 + \mathbf{x}^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$x^2 + \mathbf{x}^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$x^2 + \mathbf{x}^2 + x^2 + x^2 + x^2 + x^2 = 6\mathbf{x}^2$
$x^2 + \mathbf{x}^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$

As you can see, the alphabetic characters adapt to the selected font family but the symbols are all typeset in the same font regardless. Technically this means that the symbols are set in the fixed font family 0 whereas the alphabetic characters are typeset using variable family numbers.

Typeseting math formulas can also be done somewhat differently, as we will see in the next example.

```
$\tf\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\bf\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\sl\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\bs\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\it\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\bi\mf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
```

A new command is used: `\mf`, which stands for *math font*. This command takes care of the symbols in such a way that they are also set in the actually selected font, just like the characters.

$x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$\mathbf{x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2}$
$x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$
$x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = 6x^2$

You should take into account that TEX typesets a formula as a whole. In some cases this means that setups at the end of the formula have an effect that starts already at the beginning of the fomula.

For example, the exact location of `\mf` is not that important. We also could have typed:

```
$\bf x^2 + x^2 + x^2 + x^2 + x^2 + x^2 = \mf 6x^2$
```

There is much more to be said about math, but it is better to do that in chapter ??, about math.

1.12 Em and Ex

In specifying dimensions we can distinguish physical units like `pt` and `cm` and internal units like `em` and `ex`. These last units are related to the actual fontsize. When you use these internal units in specifying for example horizontal and vertical spacing you don't have to do any recalculating when fonts are switched in the style definition.

Some insight in these units does not hurt. The width of an `em` is not the with of an M, but that of an — (an em–dash). When this glyph is not available in the font another value is used. Table 1.14 shows some examples. We see that the width of a digit is about `.5em`. In Computer Modern Roman a digit is excactly half an em wide.

\tf	\bf	\sl	\tt	\ss	\tfx
|12|	|**12**|	|*12*|	|12|	|12|	|12|
|M|	|**M**|	|*M*|	|M|	|M|	|M|
| |	| |	| |	| |	| |	| |
⊢─┘	⊢─┘	⊢─┘	⊢---┤	⊢─┘	⊢─┘

Table 1.14 The width of an `em`.

In most cases we use `em` for specifying width and and `ex` for height. Table 1.15 shows some examples. We see that the height equals the height of a lowercase x.

| \tf	\bf	\sl	\tt	\ss	\tfx
⎯x	⎯**x**	⎯*x*	⎯x	⎯x	⎯x

Table 1.15   The height of an `ex`.

1.13 Font handling

Almost all users of typesetting systems based on TEX do so because of the quality of the output it produces. PDFTEX (and through inheritance LuaTEX as well) contains a few extensions to the typesetting engine that make the output even better than the results achieved by Knuth's original TEX. Although the extensiosn are made available by PDFTEX, they are not limited to the PDF output, they will work with the DVI backend just as well. And when the extensions are defined but not enabled, then the typeset output is 100% identical to when the feature is not present at all.

1.13.1 Character protrusion

In the following fake paragraph, you can see a hyphenation point, a secondary sentence, separated by a comma, and a last sentence, ending with a period. Miraculously, this paragraph fits into lines. Although exaggerated, these lines demonstrate that visually the hyphen and punctuation characters make the margin look ragged.
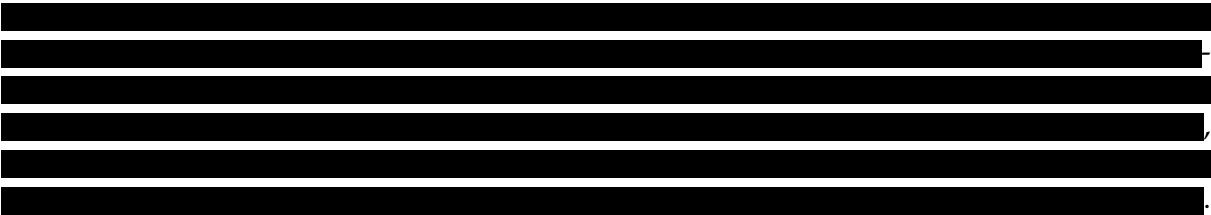
Before computers started to take over the traditional typesetter's job, it was common practice to move hyphens and punctuation into the margin, like in:

In this alternative, the margin looks less ragged, and this becomes more noticeable once you get aware of this phenomenon.

Sometimes, shifting the characters completely into the margin is too much for the sensitive eye, for instance with a slanted font, where the characters already hang to the right. In such cases, we need to compromise.

██
██-
██
██,
██
██.
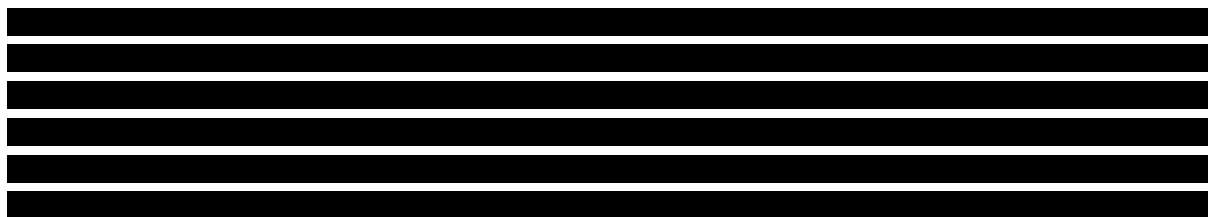
PDFTeX (and LuaTeX, that has inherited this feature) has provisions to move characters into the margin when they end up at the end of a line. Such characters are called protruding characters. PDFTeX takes protruding into account when breaking a paragraph.

We will demonstrate protruding using a quote from Hermann Zapf's article "About micro–typography and the *hz*–program" in Electronic Publishing, vol 6 (3), 1993.

```
Coming back to the use of typefaces in electronic
publishing: many of the new typographers receive their
knowledge and information about the rules of typography
from books, from computer magazines or the instruction
manuals which they get with the purchase of a PC or
software. There is not so much basic instruction, as of
now, as there was in the old days, showing the differences
between good and bad typographic design. Many people are
just fascinated by their PC's tricks, and think that a
widely||praised program, called up on the screen, will make
everything automatic from now on.
```

After TeX has typeset this paragraph (using a specific font size and line width) it may have constructed the following lines.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

As you can see, the height and depth of the lines depend on the characters, but their width equals what TeX calls `\hsize`. However, the natural width of the lines may differ from `\hsize`.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

Here the inter–word space is fixed to what TeX considers to be a space. This example also demonstrates that TeX does not have spaces, but stretches the white area between words to suit its demands. When breaking lines, TeX's mind is occupied by boxes, glue and penalties, or in more common language: (parts of) words, stretchable white space, and more or less preferred breakpoints.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

This time we have enabled PDFTEX's protruding mechanism. The characters that stick into the margin are taken into account when breaking the paragraph into lines, but in the final result, they do not count in the width. Here we used an ugly three column layout so that we got a few more hyphens to illustrate the principle, but in the next examples we will stick to two columns.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

In this first example the text was typeset in the traditional way. The hyphens and punctuation fit into the margin.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

1.13.2 Font expansion

In typesetting the two characters hz are tightly connected to Hermann Zapf and the next couple of pages we will discuss a method for optimizing the look and feel of a paragraph using a mechanism that is inspired on his work. Although official qualified in PDFTEX as font adjusting, we will use the short qualification hz since this is how it is called in the PDFTEX community.

First, here is again the same example text that was used in the previous section, typeset using normal TEX–comptibale font settings:

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

The example below shows hz in action. This paragraph is typeset with hz enabled and has a more even spacing than the text above.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

The average reader will not notice the trick, but those sensitive to character shapes will see that some glyphs are widened slightly and others are narrowed slightly. Ideally the programs that built the glyph should be defined in such a way that this goes unnoticed, but in practice glyph programs are not that clever and so a brute force horizontal scaling is applied. As long as the used percentage is small, the distortion will go unnoticed and the paragraph will look slightly better because the whitespace distribution is more even.

1.13.3 Other font handlings

In addition to the two handlings documented in the previous pararaphs (protruding and hz), ConTEXt also provides the `noligs` handling (handy when one processes xml), `flexspacing` and `prespacing` (meant for languages like French that need spacing around for instance : and ;). These handlings are experimental.

1.13.4 How to use font handlings

Before we go into the details of the actual extensions, let's see what is provided by ConTEXt as the user–level interface. The ConTEXt interface to those new features is through a subsystem called 'font handling', and at the top that subsystem is seamlessly integrated into the normal alignment macros.

For example, assuming the system is set up already to support protrusion, you can simply say

`\setupalign[hanging]`

to turn protrusion on. However, this will only work correctly if a number of special setups have taken place internally. The command `\setupalign` only toggles a switch, and the required setups have to be done elsewhere.

The list of font handling–related keys for `\setupalign` is:

`hanging`	turns on character protrusion
`nohanging`	turns off character protrusion
`hz`	turns on font expansion
`nohz`	turns off font expansion
`spacing`	turns on special spacing rules
`nospacing`	turns off special spacing

Largely because of the tight connection with the font itself, the method of defining and setting font handling is a little different between PDFTEX and MkIV.

1.13.5 Setting up font handlings in MKII

Now, let's move on to how to set up the system for font handling properly. Most of the underlying features of PDFTEX can not *just* be turned on or off, it is possible to tweak the machinery on the font as well as on the individual glyph level. You can define those settings all on your own, but ConTEXt comes with a handy set of predefined values.

name	\setupalign	description
pure	hanging	full protrusion of only selected punctuation
normal	hanging	partial protrusion of punctuation and some asymmetrical letters
hz	hz	variable correction of character widths
quality	hanging,hz	combination of hz and pure
highquality	hanging,hz	combination of hz and normal
flexspacing	spacing	automatic extra spacing around various punctuation characters
prespacing	spacing	like flexspacing, but ignoring . and , and with smaller effects
noligs	--	suppresses ligatures; because this is irreversible it is not controlled via \setupalign

You need to be aware of the fact that at the moment that you actually define a font, you need to tell what handling you want to apply.

Note: setting up font handling involves a few low-level font definition commands, so you may want to read the chapter about font definitions first.

Say that we want to hang only the serif fonts and say that we use Palatino as main typeface.

```
\setupfontsynonym [Serif] [handling=pure]
\definetypeface [palatino] [rm] [serif] [palatino] [default]
```

In the above example, the font loader is instructed to treat fonts with the virtual name `Serif` in a special way by applying the font handling named `pure`. After that, the typeface collection `palatino` is (re)defined and by that process the font tagged as `Serif` will get the 'hanging' settings attached it.

Now enable this typeface collection can be enabled by:

```
\setupbodyfont [palatino]
```

and finally, don't forget to turn on hanging by:

```
\setupalign [hanging]
```

However, this only takes care of the `Serif` font. Normally, that is the virtual name for the combination . If you also want the bold variants to hang, you have to add an extra line:

```
\setupfontsynonym [SerifBold] [handling=pure]
```

And so on for all the alternatives. This is tedious, so ConTEXt provides a shortcut. If you want to set all serif weights at once, you can call on a predefined typescript component before defining the typeface:

```
\usetypescript [serif] [handling] [pure]
```

for hanging punctuation, or for all characters:

```
\usetypescript [serif] [handling] [normal]
```

The full example then becomes:

```
\usetypescript [serif] [handling] [pure]
\definetypeface [palatino] [rm] [serif] [palatino] [default]
\setupbodyfont [palatino]
\setupalign [hanging]
```

The first argument can be one of three named typescript groups: `serif` (for the virtual font synonyms whose names begin with `Serif`), `sans` (for `Sans`), or `mono` (for `Mono`). The second argument should always be `handling`. The third argument has to be one of named font handlings that are listed in the table at the start of this section.

The typescripts that are used in these examples work by altering the font synonyms for virtualized symbolic font names like `Serif` and `SerifBold` en block. They will even work with your own typescripts if (but only if) these typescripts use the same font naming conventions as the ConTEXt core.

The definition of font handlings is actually a two-step process. A named font handling consists of one or more handling vectors that have to be defined first, those are then combined under a single name.

This is not the right place to describe how to define the low-level vector definitions in detail, for that you are referred to the documented source of the main handling definition file `hand-def.tex`. But to give you an idea of what it looks like, here is a small excerpt of that file. The `pure` handling vector is defined as:

```
\startfonthandling [pure]

  \defineprotrudefactor , 0 1
  \defineprotrudefactor . 0 1
  \defineprotrudefactor : 0 1
  \defineprotrudefactor ; 0 1
  \defineprotrudefactor - 0 1

  \defineprotrudefactor hyphen 0 1
  \defineprotrudefactor endash 0 .5
  \defineprotrudefactor emdash 0 .33 % .5

\stopfonthandling
```

The `pure` font handling itself is then defined as follows:

```
\definefonthandling [pure] [pure] [type=hanging]
```

The `hz` setup runs along the same lines. First here is a vector:

```
\startfonthandling [hz]

  \defineadjustfactor A .5
  \defineadjustfactor B .7
  \defineadjustfactor C .7
  ...

\stopfonthandling
```

And then the definition of the hz handling is as follows:

```
\definefonthandling [hz] [hz,extended] [type=hz]
```

To wrap this up, here is the macro syntax for the font handling definition and setup.

```
\definefonthandling [.¹.] [.².] [.³.]

1   IDENTIFIER

2   IDENTIFIER

3   type    =   hanging  hz  spacing  tag
    right   =   NUMBER
    left    =   NUMBER
    factor  =   NUMBER
    min     =   NUMBER
    max     =   NUMBER
    step    =   NUMBER
```

As you can see, the \definefonthandling command accepts three arguments. The first is the handling to be defined, the second is a list of handling vectors to be used, and the third sets up a number of settings.

type   the type of this font handling feature, for use by \setupalign
right   used by type=hanging, default 1
left   used by type=hanging, default 1
factor   used by type=spacing, default 1
min   used by type=hz, default 20
max   used by type=hz, default 20
step   used by type=hz, default 5

On top of the list at the beginning of this paragraph, a few more elaborate font handlings are also predefined:

```
\definefonthandling [purebold]          [pure] [type=hanging]
\definefonthandling [pureslanted]       [pure] [type=hanging,right=1.5]
\definefonthandling [pureitalic]        [pure] [type=hanging,right=1.5]
\definefonthandling [pureboldslanted]   [pure] [type=hanging,right=1.5]
\definefonthandling [purebolditalic]    [pure] [type=hanging,right=1.5]
```

The right parameter (there is also left) is a multiplication factor that is applied to the values in the associated vector. Such definitions can be more extensive, like:

```
\definefonthandling
  [normalitalic]
  [punctuation,alpha,extended]
  [type=hanging,right=1.5]
```

Here we have combined three vectors into one handling. For these extended font handlings, there are no predefined typescripts, so you either have to use the font synonyms directly, or define your own typescripts. Now, if you think this is overly complicated, you are probably right. Normally you will just invoke protruding handlings defined previously, but the mechanisms are there to fine–tune the handlings to your precise wishes.

In case you want to alter some of the settings of an already defined font handling, there is

```
\setupfonthandling [.¹.] [.².]

1   IDENTIFIER

2   inherits from \definefonthandling
```

The first argument is the handling to be altered, the second sets up the settings.

1.13.6   Setting up font handlings in MKIV

In MkIV, font handling is merged with the font features (because these already have a low-level connection to the font), so you can set up the font-side of things with the sixth argument of \definetypeface, like so:

```
\definefontfeature
    [hz] [default]
```

```
    [protrusion=pure, mode=node, script=latn]
\definetypeface [palatino] [rm] [serif] [palatino] [default] [features=hz]
\setupbodyfont [palatino]
\setupalign [hanging]
```

or by redefining the feature set that is used by the typescript you are using and then (re-)exe-
cuting the typescript, like so:

```
\definefontfeature
    [default] [default]
    [protrusion=pure, expansion=quality, mode=node, script=latn]
\usetypescript[palatino]
\setupbodyfont [palatino]
\setupalign [hanging]
```

There is a list of predefined font handling feature values that you can use:

For protrusion, there is:

name	\setupalign	description
pure	hanging	full protrusion of only selected punctuation
punctuation	hanging	partial protrusion of punctuation
alpha	hanging	partial of some asymmetrical letters
quality	hanging	the combination of punctuation and alpha

For expansion, there is:

name	\setupalign	description
quality	hz	variable correction of character widths

These are defined in the file font-ext.lua. The low–level definitions look like

```
fonts.protrusions.vectors['pure'] = {
    [0x002C] = { 0, 1    }, -- comma
    [0x002E] = { 0, 1    }, -- period
    [0x003A] = { 0, 1    }, -- colon
    [0x003B] = { 0, 1    }, -- semicolon
    [0x002D] = { 0, 1    }, -- hyphen
    [0x2013] = { 0, 0.50 }, -- endash
    [0x2014] = { 0, 0.33 }, -- emdash
}
fonts.protrusions.classes['pure'] = {
    vector = 'pure', factor = 1
}
```

That was the complete definition of protrusion=pure. The key classes has the same func-
tion as the macro call \definefonthandling in MkII. It references the named vector pure and
sets up a parameter.

For protrusion, there is only the one parameter factor, but for expansion there are a few
more:

```
fonts.expansions.classes['quality'] = {
    stretch = 2, shrink = 2, step = .5, vector = 'default', factor = 1
```

```
}
fonts.expansions.vectors['default'] = {
    [byte('A')] = 0.5,
    [byte('B')] = 0.7,
    ... -- many more characters follow
}
```

As you can see, the definition order of vector vs. class is not important, and the format of the vector is a little different. The use of `byte()` is just so that that keying in hex numbers can be avoided. The values are bare numbers instead of hashes because there is only one per-character parameter involved with character expansion.

Also note that the values for the parameters `stretch`, `shrink` and `step` are divided by a factor 10 compared to the MkII definition.

IN MkIV, there is no support for the `spacing` key to `\setupalign` yet. That is because the low–level features in PDFTEX are not present in LuaTEX, and there is no replacement yet. The font handling `noligs` is, of course, replaced by the OpenType font feature tags for ligatures: simply leave all of the relevant font features turned off.

1.14 Encodings and mappings

This paragraph only applies to PDFTEX. If you are exclusively using XƎTEX or MkIV, you can safely ignore the following text.

Not every language uses the (western) latin alphabet. Although in most languages the basic 26 characters are somehow used, they can be combined with a broad range of accents placed in any place.

In order to get a character representation, also called glyph, in the resulting output, you have to encode it in the input. This is no problem for `a..z`, but other characters are accessed by name, for instance `\eacute`. The glyph é can be present in the font but when it's not there, TEX has to compose the character from a letter e and an accent `.

In practice this means that the meaning of `\eacute` depends on the font and font encoding used. There are many such encodings, each suited for a subset of languages.

encoding	usage	status
8r	a (strange) mixture of encodings	useless
default	the 7 bit ASCII encoding as used by plain TEX	obsolete
ec	the prefered encoding of TEX distributions	okay
greek	an encoding for modern greek	okay
qx	an encoding that covers most eastern european languages	okay
t2a	a cyrillic TEX font encoding	?
t2b	another cyrillic TEX font encoding	?
t2c	another another cyrillic TEX font encoding	?
t5	an encoding dedicated to vietnamese (many (double) accents)	okay
texnansi	a combination of TEX and Adobe standard encoding	okay

These encodings are font related as is demonstrated in figure 1.1, 1.2, 1.3, and 1.4. Here we used the `\showfont` command.

Figure 1.1   The Latin Modern Roman font in ec encoding.

The situation is even more complicated than it looks, since the font may be virtual, that is, built from several fonts.

The advantage of using specific encodings is that you can let TeX hyphenate words in the appropriate way. The hyphenation patterns are applied to the internal data structures that represent the sequence of glyphs. In spite of what you may expect, they are font–dependent! Even more confusing: they not only depend on the font encoding, but also on the mapping from lower to uppercase characters, or more precise, on the existence of such a mapping.

Unless you want to play with these encodings and mappings, in most cases you can forget their details and rely on what other TeX experts tell you to do. Normally switching from one to another encoding and/or mapping takes place with the change in fonts or when some special

name: lmr10 at 11.0pt   encoding: default   mapping: default

Figure 1.2   The Latin Modern Roman font in texnansi encoding.

output encoding is needed, for instance in PDF annotations and/or unicode vectors that enable searching in documents. So, to summarize this: encodings and mappings depend on the fonts used as well have consequences for the language specific hyphenation patterns. Fortunately CONTEXT handles this for you automatically.

If you want to know to what extent a font is complete and characters need to be composed on the fly, you can typeset a a couple of tables. The (current) composition is shown by \showaccents:

```
——— default /opt/tex/texmf-local/fonts/data/public/tex-gyre/texgyrepagella-regular.ot
\'   á b́ ć d́ é f́ ǵ h́ í j́ ḱ Ĺ ḿ ń ó ṕ q́ ŕ ś t́ ú v́ ẃ x́ ý ź
     Á B́ Ć D́ É F́ Ǵ H́ Í J́ Ḱ Ĺ Ḿ Ń Ó Ṕ Q́ Ŕ Ś T́ Ú V́ Ẃ X́ Ý Ź
\`   à b̀ c̀ d̀ è f̀ g̀ h̀ ì j̀ k̀ Ì m̀ ǹ ò p̀ q̀ r̀ s̀ t̀ ù v̀ ẁ x̀ ỳ z̀
```

Figure 1.3 The Latin Modern Roman font in qx encoding.

`name: lmr10 at 11.0pt`   `encoding: default`   `mapping: default`

	À	Ḃ	Ċ	Ḋ	È	Ḟ	Ġ	Ḣ	Ì	J̇	K̇	L̇	Ṁ	Ṅ	Ò	Ṗ	Q̇	Ṙ	Ṡ	Ṫ	Ù	V̇	Ẇ	Ẋ	Ẏ	Ż
\^	â	b̂	ĉ	d̂	ê	f̂	ĝ	ĥ	î	ĵ	k̂	l̂	m̂	n̂	ô	p̂	q̂	r̂	ŝ	t̂	û	v̂	ŵ	x̂	ŷ	ẑ
	Â	B̂	Ĉ	D̂	Ê	F̂	Ĝ	Ĥ	Î	Ĵ	K̂	L̂	M̂	N̂	Ô	P̂	Q̂	R̂	Ŝ	T̂	Û	V̂	Ŵ	X̂	Ŷ	Ẑ
\~	ã	b̃	c̃	d̃	ẽ	f̃	g̃	h̃	ĩ	j̃	k̃	l̃	m̃	ñ	õ	p̃	q̃	r̃	s̃	t̃	ũ	ṽ	w̃	x̃	ỹ	z̃
	Ã	B̃	C̃	D̃	Ẽ	F̃	G̃	H̃	Ĩ	J̃	K̃	L̃	M̃	Ñ	Õ	P̃	Q̃	R̃	S̃	T̃	Ũ	Ṽ	W̃	X̃	Ỹ	Z̃
\"	ä	b̈	c̈	d̈	ë	f̈	g̈	ḧ	ï	j̈	k̈	l̈	m̈	n̈	ö	p̈	q̈	r̈	s̈	ẗ	ü	v̈	ẅ	ẍ	ÿ	z̈
	Ä	B̈	C̈	D̈	Ë	F̈	G̈	Ḧ	Ï	J̈	K̈	L̈	M̈	N̈	Ö	P̈	Q̈	R̈	S̈	T̈	Ü	V̈	Ẅ	Ẍ	Ÿ	Z̈
\H	á̋	b̋	c̋	d̋	e̋	f̋	g̋	h̋	i̋	j̋	k̋	l̋	m̋	n̋	ő	p̋	q̋	r̋	s̋	t̋	ű	v̋	w̋	x̋	y̋	z̋
	A̋	B̋	C̋	D̋	E̋	F̋	G̋	H̋	I̋	J̋	K̋	L̋	M̋	N̋	Ő	P̋	Q̋	R̋	S̋	T̋	Ű	V̋	W̋	X̋	Y̋	Z̋
\r	å	b̊	c̊	d̊	e̊	f̊	g̊	h̊	i̊	j̊	k̊	l̊	m̊	n̊	o̊	p̊	q̊	r̊	s̊	t̊	ů	v̊	ẘ	x̊	ẙ	z̊

name: lmr10 at 11.0pt   encoding: default   mapping: default

Figure 1.4   The Latin Modern Roman font in t5 encoding.

	Å	B̊	Č	Ď	E̊	F̌	Ǧ	Ḣ	İ	J̌	Ǩ	Ľ	M̌	Ň	O̊	P̊	Q̊	Ř	Š	Ť	Ů	V̌	W̊	X̊	Ŷ	Ż
\v	ă	b̆	č	ď	ě	f̌	ǧ	ȟ	ǐ	ǰ	ǩ	ľ	m̌	ň	ǒ	p̌	q̌	ř	š	ť	ǔ	v̌	w̌	x̌	y̌	ž
	Ǎ	B̌	Č	Ď	Ě	F̌	Ǧ	Ȟ	Ǐ	J̌	Ǩ	Ľ	M̌	Ň	Ǒ	P̌	Q̌	Ř	Š	Ť	Ǔ	V̌	W̌	X̌	Y̌	Ž
\u	ă	b̆	c̆	d̆	ĕ	f̆	ğ	h̆	ĭ	j̆	k̆	l̆	m̆	n̆	ŏ	p̆	q̆	r̆	s̆	t̆	ŭ	v̆	w̆	x̆	y̆	z̆
	Ă	B̆	C̆	D̆	Ĕ	F̆	Ğ	H̆	Ĭ	J̆	K̆	L̆	M̆	N̆	Ŏ	P̆	Q̆	R̆	S̆	T̆	Ŭ	V̆	W̆	X̆	Y̆	Z̆
\=	ā	b̄	c̄	d̄	ē	f̄	ḡ	h̄	ī	j̄	k̄	l̄	m̄	n̄	ō	p̄	q̄	r̄	s̄	t̄	ū	v̄	w̄	x̄	ȳ	z̄
	Ā	B̄	C̄	D̄	Ē	F̄	Ḡ	H̄	Ī	J̄	K̄	L̄	M̄	N̄	Ō	P̄	Q̄	R̄	S̄	T̄	Ū	V̄	W̄	X̄	Ȳ	Z̄
\.	ȧ	ḃ	ċ	ḋ	ė	ḟ	ġ	ḣ	i̇	j̇	k̇	l̇	ṁ	ṅ	ȯ	ṗ	q̇	ṙ	ṡ	ṫ	u̇	v̇	ẇ	ẋ	ẏ	ż
	Ȧ	Ḃ	Ċ	Ḋ	Ė	Ḟ	Ġ	Ḣ	İ	J̇	K̇	L̇	Ṁ	Ṅ	Ȯ	Ṗ	Q̇	Ṙ	Ṡ	Ṫ	U̇	V̇	Ẇ	Ẋ	Ẏ	Ż
\b	a̲	b̲	c̲	d̲	e̲	f̲	g̲	h̲	i̲	j̲	k̲	l̲	m̲	n̲	o̲	p̲	q̲	r̲	s̲	t̲	u̲	v̲	w̲	x̲	y̲	z̲

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
\d	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
\k	ą	b	c	d	ę	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	ų	v	w	x	y	z
	Ą	B	C	D	Ę	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	Ų	V	W	X	Y	Z
\c	a	b	ç	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	ş	ţ	u	v	w	x	y	z
	A	B	Ç	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Ş	Ţ	U	V	W	X	Y	Z

with \showcharacters, you get a list of named characters (and glyphs) as known to the system.

```
——— default /opt/tex/texmf-local/fonts/data/public/tex-gyre/texgyrepagella-regular.ot
```

,	textcomma	¿	questiondown	†	textdag
.	textperiod	©	copyright	‡	textddag
´	textacute	®	registered	°	textdegree
.	textbottomdot	™	trademark	÷	textdiv
˘	textbreve	§	sectionmark	…	textellipsis
ˇ	textcaron	¶	paragraphmark	/	textfraction
¸	textcedilla	¼	onequarter	¬	textlognot
^	textcircumflex	½	onehalf		textminus
¨	textdiaeresis	¾	threequarter	µ	textmu
˙	textdotaccent	¹	onesuperior	×	textmultiply
`	textgrave	²	twosuperior	±	textpm
˝	texthungarumlaut	³	threesuperior	"	quotedbl
¯	textmacron	¢	textcent	„	quotedblbase
˛	textogonek	¤	textcurrency	"	quotedblleft
˚	textring	$	textdollar	"	quotedblright
˜	texttilde	€	texteuro	`	quotesingle
,	textbottomcomma	ƒ	textflorin	,	quotesinglebase
ı	dotlessi	£	textsterling	'	quoteleft
	dotlessj	¥	textyen	'	quoteright
I	dotlessI	ª	ordfeminine	‹	guilsingleleft
J	dotlessJ	º	ordmasculine	›	guilsingleright
–	endash	%	percent	«	leftguillemot
—	emdash	‰	perthousand	»	rightguillemot
æ	aeligature	-	softhyphen	Â	Acircumflex
Æ	AEligature	·	periodcentered	â	acircumflex
ij	ijligature		compoundwordmark	Ĉ	Ccircumflex
IJ	IJligature	^	textasciicircum	ĉ	ccircumflex
œ	oeligature	~	textasciitilde	Ê	Ecircumflex
Œ	OEligature	/	textslash	ê	ecircumflex
ß	ssharp	\	textbackslash	Ĝ	Gcircumflex
SS	Ssharp	{	textbraceleft	ĝ	gcircumflex
þ	thorn	}	textbraceright	Ĥ	Hcircumflex
Þ	Thorn		textunderscore	ĥ	hcircumflex
■	eth	␣	textvisiblespace	Î	Icircumflex
Ð	Eth	¦	textbrokenbar	î	icircumflex
¡	exclamdown	•	textbullet	Ĵ	Jcircumflex

ĵ	jcircumflex	Á	Aacute	ō	omacron
Ô	Ocircumflex	á	aacute	Ū	Umacron
ô	ocircumflex	Ć	Cacute	ū	umacron
Ŝ	Scircumflex	ć	cacute	Ç	Ccedilla
ŝ	scircumflex	É	Eacute	ç	ccedilla
Û	Ucircumflex	é	eacute	Ķ	Kcedilla
û	ucircumflex	Í	Iacute	ķ	kcedilla
Ŵ	Wcircumflex	í	iacute	Ļ	Lcedilla
ŵ	wcircumflex	Ĺ	Lacute	ļ	lcedilla
Ŷ	Ycircumflex	ĺ	lacute	Ņ	Ncedilla
ŷ	ycircumflex	Ń	Nacute	ņ	ncedilla
À	Agrave	ń	nacute	Ŗ	Rcedilla
à	agrave	Ó	Oacute	ŗ	rcedilla
È	Egrave	ó	oacute	Ş	Scedilla
è	egrave	Ŕ	Racute	ş	scedilla
Ì	Igrave	ŕ	racute	Ţ	Tcedilla
ì	igrave	Ś	Sacute	ţ	tcedilla
Ò	Ograve	ś	sacute	Ő	Ohungarumlaut
ò	ograve	Ú	Uacute	ő	ohungarumlaut
Ù	Ugrave	ú	uacute	Ű	Uhungarumlaut
ù	ugrave	Ý	Yacute	ű	uhungarumlaut
Ỳ	Ygrave	ý	yacute	Ą	Aogonek
ỳ	ygrave	Ź	Zacute	ą	aogonek
Ã	Atilde	ź	zacute	Ę	Eogonek
ã	atilde	đ	dstroke	ę	eogonek
Ĩ	Itilde	Đ	Dstroke	Į	Iogonek
ĩ	itilde	Ħ	Hstroke	į	iogonek
Ñ	Ntilde	ħ	hstroke	Ų	Uogonek
ñ	ntilde		Tstroke	ų	uogonek
Õ	Otilde		tstroke	Å	Aring
õ	otilde	Ċ	Cdotaccent	å	aring
Ũ	Utilde	ċ	cdotaccent	Ů	Uring
ũ	utilde	Ė	Edotaccent	ů	uring
Ỹ	Ytilde	ė	edotaccent	Ă	Abreve
ỹ	ytilde	Ġ	Gdotaccent	ă	abreve
Ä	Adiaeresis	ġ	gdotaccent	Ĕ	Ebreve
ä	adiaeresis	İ	Idotaccent	ĕ	ebreve
Ë	Ediaeresis	i	idotaccent	Ğ	Gbreve
ë	ediaeresis	Ż	Zdotaccent	ğ	gbreve
Ï	Idiaeresis	ż	zdotaccent	Ĭ	Ibreve
ï	idiaeresis	Ā	Amacron	ĭ	ibreve
Ö	Odiaeresis	ā	amacron	Ŏ	Obreve
ö	odiaeresis	Ē	Emacron	ŏ	obreve
Ü	Udiaeresis	ē	emacron	Ŭ	Ubreve
ü	udiaeresis	Ī	Imacron	ŭ	ubreve
Ÿ	Ydiaeresis	ī	imacron	Č	Ccaron
ÿ	ydiaeresis	Ō	Omacron	č	ccaron

Ď	Dcaron	ẻ	ehook	ạ	adotbelow
ď	dcaron	Ỉ	Ihook	Ẹ	Edotbelow
Ě	Ecaron	ỉ	ihook	ẹ	edotbelow
ě	ecaron	Ỏ	Ohook	Ị	Idotbelow
Ľ	Lcaron	ỏ	ohook	ị	idotbelow
ľ	lcaron	Ủ	Uhook	Ọ	Odotbelow
Ň	Ncaron	ủ	uhook	ọ	odotbelow
ň	ncaron	Ỷ	Yhook	Ụ	Udotbelow
Ř	Rcaron	ỷ	yhook	ụ	udotbelow
ř	rcaron	Ầ	Acircumflexgrave	Ỵ	Ydotbelow
Š	Scaron	Ấ	Acircumflexacute	ỵ	ydotbelow
š	scaron	Ẫ	Acircumflextilde	Ợ	Ohorndotbelow
Ť	Tcaron	Ẩ	Acircumflexhook	ợ	ohorndotbelow
ť	tcaron	ầ	acircumflexgrave	Ự	Uhorndotbelow
Y̌	Ycaron	ấ	acircumflexacute	ự	uhorndotbelow
y̌	ycaron	ẫ	acircumflextilde	Ậ	Acircumflexdotbelow
Ž	Zcaron	ẩ	acircumflexhook	ậ	acircumflexdotbelow
ž	zcaron	Ề	Ecircumflexgrave	Ệ	Ecircumflexdotbelow
Ł	Lstroke	Ế	Ecircumflexacute	ệ	ecircumflexdotbelow
ł	lstroke	Ễ	Ecircumflextilde	Ộ	Ocircumflexdotbelow
Ø	Ostroke	Ể	Ecircumflexhook	ộ	ocircumflexdotbelow
ø	ostroke	ề	ecircumflexgrave	Ặ	Abrevedotbelow
ä	aumlaut	ế	ecircumflexacute	ặ	abrevedotbelow
ë	eumlaut	ễ	ecircumflextilde	Ơ	Ohorn
ï	iumlaut	ể	ecircumflexhook	Ờ	Ohorngrave
ö	oumlaut	Ồ	Ocircumflexgrave	Ớ	Ohornacute
ü	uumlaut	Ố	Ocircumflexacute	Ỡ	Ohorntilde
Ä	Aumlaut	Ỗ	Ocircumflextilde	Ở	Ohornhook
Ë	Eumlaut	Ổ	Ocircumflexhook	ơ	ohorn
Ï	Iumlaut	ồ	ocircumflexgrave	ờ	ohorngrave
Ö	Oumlaut	ố	ocircumflexacute	ớ	ohornacute
Ü	Uumlaut	ỗ	ocircumflextilde	ỡ	ohorntilde
ş	scommaaccent	ổ	ocircumflexhook	ở	ohornhook
Ş	Scommaaccent	Ằ	Abrevegrave	Ư	Uhorn
ţ	tcommaaccent	Ắ	Abreveacute	Ừ	Uhorngrave
Ţ	Tcommaaccent	Ẵ	Abrevetilde	Ứ	Uhornacute
ļ	lcommaaccent	Ẳ	Abrevehook	Ữ	Uhorntilde
Ļ	Lcommaaccent	ằ	abrevegrave	Ử	Uhornhook
Ẽ	Etilde	ắ	abreveacute	ư	uhorn
ẽ	etilde	ẵ	abrevetilde	ừ	uhorngrave
Ả	Ahook	ẳ	abrevehook	ứ	uhornacute
ả	ahook	Ạ	Adotbelow	ữ	uhorntilde
Ẻ	Ehook			ử	uhornhook

If you want to know what patterns are used, you can try to hyphenate a word with `\showhy-phenations`.

language : en (internal code:2)
font   : /opt/tex/texmf-local/fonts/data/public/tex-gyre/texgyrepagella-regular.otf at 11.0pt
encoding : not set
mapping : not set
handling : not set
sample   : abra-cadabra