



Hans Hagen
Taco Hoekwater

fonts in context

Colofon

This manual is typeset with ConT_EXt and METAPOST. No special tricks are used and everything you see in here, is available for ConT_EXt users. The text is typeset in Linux Libertine and Latin Modern Typewriter. We used LuaT_EX as T_EX processing engine.

Copyright

Hans Hagen, PRAGMA Advanced Document Engineering, Hasselt NL
Taco Hoekwater, Bittext, Dordrecht NL
copyright: 1999-2011

Publisher

publisher: Boekplan, NL
isbn-ean: 978-94-90688-03-5
website: www.boekplan.nl

Info

internet: www.pragma-ade.com
support: ntg-context@ntg.nl
email: pragma@wxs.nl

Introduction

ConTeXt is a document markup language and document preparation system based on the TeX typesetting system. It was designed with the same general-purpose aims as L^ATeX: providing an easy to use interface to the high quality typesetting engine TeX. However, while L^ATeX insulates the writer from typographical details, ConTeXt takes a complementary approach by providing structured interfaces for handling typography, including extensive support for colors, backgrounds, hyperlinks, presentations, figure/text integration, and conditional compilation. It gives the user extensive control over formatting while making it easy to create new layouts and styles without learning the TeX macro language.

The current development version of ConTeXt is labeled ‘MKIV’, and runs on the TeX-derived typesetting engine called LuaTeX. For this version, many parts of ConTeXt have been redesigned from scratch making extensive use of the Lua programming language. And that is besides the heavy use of LuaTeX features like support for OpenType and TrueType fonts and support for Unicode input.

Over the past years, a number of articles have been published by Hans Hagen about various parts of the new font subsystem in ConTeXt MKIV, but this book is the first attempt to combine all information relating to fonts in a single document. I did my best to make sure that on the one hand all information that is needed is actually given while at the other hand attempting to stay away from the nitty-gritty details. ConTeXt MKIV is still a work in progress, and sometimes things change, especially at the lowest level of detail.

This book is about fonts in ConTeXt, and as such it assumes that some knowledge about ConTeXt itself is already present. No attempt is made to explain the basics of creating input files or running ConTeXt: if you are completely new to ConTeXt, it makes sense to study ‘ConTeXt, an excursion’ first. You can download ‘ConTeXt, an excursion’ from the Pragma ADE website at <http://www.pragma-ade.com>, or find it using the excellent ConTeXt community wiki at <http://wiki.contextgarden.net>. The latter is also a very good starting point for learning about other ConTeXt-related topics.

This book could not have been written within a reasonable time frame without the already existing articles by Hans Hagen, the articles on the wiki, and the replies on the ConTeXt mailing list to fall back upon. I want to extend a very heartfelt ‘thank you!’ to all contributors that I somewhat sneakily stole text from. And if there are any errors in this book, blame me. Better still: tell me about them, so that they can be fixed in a future update.

Breskens, February 2011

Taco Hoekwater

Table of Contents

1	Before you begin . . .	9
1.1	Typography	9
1.2	The ConT _E Xt font mechanism	12
2	Font switching	13
2.1	Font style switching	13
2.2	Font alternative switching	14
2.3	Switching font styles in setup commands	15
2.4	Emphasize	16
2.5	Line spacing	17
2.6	Capitals	20
2.7	Character spacing	23
2.8	Underlining text	23
2.9	Selecting body fonts	24
2.10	Body font sizes	24
2.11	Body font environments	25
2.12	Displaying the current font setup	27
2.13	Symbols	29
2.14	Math fonts	32
2.15	Em and Ex units	33
2.16	Simple font switching	34
3	Paragraph alignment	37
3.1	Character protrusion	37
3.2	Font expansion	40
3.3	How to use paragraph alignment	40
4	Font names	43
4.1	Introduction	43
4.2	Method 1: file	44
4.3	Method 2: name	45
4.4	Method 3: spec	47
4.5	The font database	48
4.6	Interfacing	49
4.7	A few remarks	51

5	Single fonts and symbols	53
5.1	Simple font definitions	53
5.2	About font encodings	54
5.3	Loading fonts at sizes	55
5.4	Font synonyms	57
5.5	Showing fonts	58
5.6	Defining symbols	60
6	Selecting document fonts	65
6.1	Body font identifiers	65
6.2	Typescript files	67
6.3	Predefined typescripts	69
7	Body font definitions	73
7.1	Introduction	73
7.2	Standard font synonyms	73
7.3	Defining body fonts	73
8	Typescripts and typefaces	81
8.1	A simple example	81
8.2	Some internals	84
8.3	A typescript in action	85
8.4	Some more details	86
9	Font features	91
9.1	Defining font feature sets	91
9.2	Setting up font handling	94
9.3	Glyph composition	95
10	Font Fallbacks	97
11	Font Goodies	101
11.1	Introduction	101
11.2	Color	106
11.3	The goodies file	108
11.4	Optimizing Arabic	110
11.5	Protrusion and expansion	114
12	Extending the system	115
12.1	Predefined font, style and alternative keywords	115
12.2	Defining font styles	117
12.3	Defining font alternatives	118
13	Installing fonts	119
13.1	System fonts	119
13.2	Texmf fonts	120

1 Before you begin . . .

1.1 Typography

Throughout the millennia humans have developed and adapted methods for storing facts and thoughts on a variety of different media. A very efficient way of doing this is using logograms, as the Chinese have done for ages. Another method is to represent each syllable in a word by a symbol. A third method is by using a limited set of shapes representing basic sounds (a.k.a. phonemes). Such a collection is called an alphabet, and the separate shapes are called letters. The components of written language like logograms, syllables, and letters can be grouped under a single word. We call these *characters*.

The shapes representing the characters are more or less standardized, and thereby can be recognized by readers even if their details differ. A collection of pictures representing such shapes is called a *font*, and the separate pictures in a font are called *glyphs*.

Usually, one glyph in a font represents one character. But there are exceptions: it is possible that there is a single glyph that combines the representation of multiple characters, and it is equally possible that there are multiple glyphs in a font for a single character. Because of these possibilities, it is important to remember the distinction between characters (the core units of language) and glyphs (physical representations of language). Naturally, the rest of this book will be more about glyphs than about characters.

The example below shows (from left to right) a Computer Modern font, a Helvetica lookalike, a Times Roman lookalike and the Antiqua Torunska font.

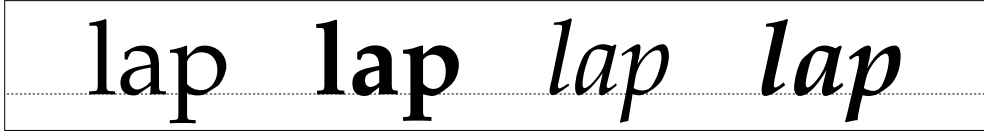


As you can see, quite some design variation is possible. It follows that when fonts from different sources (designers) are intermixed, the result is not always pleasing to look at. The term *font collection* refers to a set of fonts combined together in such a way that the overall appearance on a page looks good and reading is as comfortable as possible.

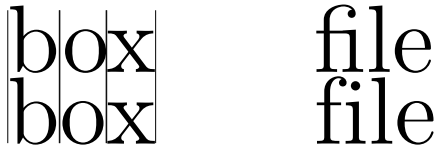
The next example shows an attempt at such a font collection: the fonts were picked such that the glyph sizes and the line thicknesses are roughly the same.



Fonts from a single source often already come in sets containing a few variations that are intended to be used together. Such a set of fonts with the same basic design is known as a *font family*. In the example below there are a normal, a bold, an italic, and a bold italic *alternative* of a font.



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*.



The font shown here is Computer Modern, the default \TeX font. This font was created by Donald Knuth. The Computer Modern has many kerning pairs, while the font used for most of the text in this manual has relatively few, while both have essentially the same list of ligatures.

Micro-typography features like kerning and ligatures are not to be altered by the user, but are part of the font design and the required information is stored inside the font file, together with the drawing routines for the actual pictures. It is possible for the user to alter which fonts are used, the interline spacing, and some more aspects on the level of macro-typography. The choice of font is the main topic of this book.

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; on the font's intended application, like a newspaper or a book. Often, classification systems mix these characteristics to a certain point.

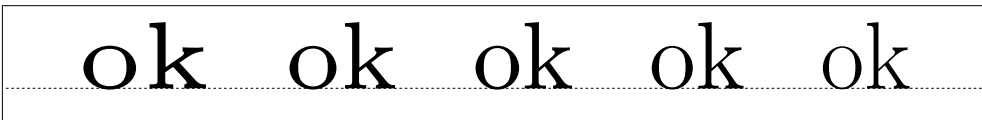
For example, the Computer Modern family can be classified as a 'modern' font. This is a classification that primarily indicates a period (late 18th century), but it also implies a particular shape: 'modern' fonts have a high contrast between thick and thin strokes, and their stress axis is perfectly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

At the same time, specific fonts in the Computer Modern family can be classified as 'serif' (glyph strokes have embellishments at the end), 'sans serif' (shapes end abruptly), or 'mono-spaced' (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 each other in style. In ConT_EXt you will normally use a reimplementaion 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.



In the example above 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. Traditionally, font designers used to design a glyph collection for each point size, but nowadays most fonts have only a single design size of 10 points, or at most a small set of sizes with names indicating their proposed use, like *caption*, *text*, and *display*.



As explained earlier, the general appearance of a font style can be classified according to many schemes, and the exact terminology used depends on the background of the user. For example, typographers and book designers will often talk about ‘main’, ‘support’, and ‘monospaced’ fonts, whereas T_EXies usually use ‘roman’, ‘sans’, and ‘typewriter’ for the same font style groups.

In ConT_EXt many such terms can be used intermixed because they are all remapped to the same set of internal commands. As will be explained later, the command `\rm` is used to switch to the style used for the main text (this is usually a font style with serifs), `\ss` to switch to the support style (usually a style without serifs) and `\tt` to switch to the code example style (for which usually monospaced fonts are used).

Text can be typeset in different font sizes. The unit `pt`, short for ‘printer’s point’, is normally used to specify the size of a font. There are a little over 72 points per inch (or a little under 2.85 points per millimeter, if you prefer metric units).

The next chapters will go into the details of switching font styles and fonts in your documents. Be warned that the font switching mechanism is rather complex. This is due to the different modes like math mode and text mode in ConT_EXt. If you want to understand the mechanism fully, you will have to obtain some knowledge on fonts and their peculiarities.

1.2 The ConT_EXt font mechanism

Font switching is one of the oldest features of ConT_EXt because font switching is indispensable in a macro package. As the years passed, extensions to the font switching mechanism were inevitable. The following starting points have been chosen during the development of the mechanism:

- It must be easy to change *styles*, e.g., switching between roman (serif, regular), sans serif (support), teletype (monospaced), et cetera.
- More than one *alternative* set of glyph shapes must be available, like italic and bold.
- Different font families like Latin Modern Roman and Lucida Bright must be supported.
- It must be possible to combine different font styles into font collections. Such collections are called *typescripts*, and in there constituent styles are called *typefaces*.
- Different sub- and super-script sizes must be available. These script sizes have to be consistent across the switching of typeface, style and alternative.
- It should be possible to combine all of these requirements into a single definition unit.
- 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. ConT_EXt's terminology probably differs from what you are accustomed to, especially if you were previously a L^AT_EX user. What L^AT_EX calls a 'font family', is named 'font style' or 'typeface' in ConT_EXt, and what L^AT_EX calls a 'font style' is called 'font alternative' in ConT_EXt.

2 Font switching

A quick recap: we call a collection of fonts, like Lucida or Computer Modern Roman, a *collection*. Within such a collection, the members can be grouped according to characteristics. Such a group is called a *style* or *typeface*. 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 ‘italic’.

There are different ways to change to a new 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 mostly typeset using just one combination of family and style. This combination is referred to as the ‘body font’ of the document.

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

2.1 Font style switching

Switching to another font style is done by one of five two-letter commands that are listed in table 2.1.

command	Long names and aliases
<code>\rm</code>	serif, regular, roman, rm
<code>\ss</code>	sans, support, sansserif, ss
<code>\tt</code>	mono, type, teletype, tt
<code>\hw</code>	handwritten, hw
<code>\cg</code>	calligraphic, cg
–	mm

Table 2.1 Font style switching commands and their keyword equivalents.

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, as will be explained in one of the following chapters.

There is a sixth internal style that is only referred to 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, and it is only mentioned here for completeness.

2.2 Font alternative switching

The possible alternatives within a style are given in table 2.2. In the real world, not all fonts have both italic and slanted or the bold alternatives of each. Some other fonts do not have small caps or have 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 if slanted is not available or vice versa.

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

Table 2.2 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, and you use `\tf` to go back to the default alternative for the current font style.

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, . . . selects a somewhat larger actual font from the previous one. The x and xx suffixes select smaller and even smaller versions.

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

Table 2.3 Small alternative switching commands and their keyword equivalents.

The ‘small’ switches mentioned in table 2.3 are always available. The availability of other commands like `\ita`, `\bfxx`, `\bfc`, etc. depends on the completeness of the font definition

files. For the core ConT_EXt 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 character size, 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 2.11.

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, ConT_EXt 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 section is for the sake of completeness. Using 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 by combining a style and an alternative in a single control sequence: `\rmsl`, `\ssbf`, `\tttfa`, et cetera.

2.3 Switching font styles in setup commands

A number of ConT_EXt commands use the parameter `style` to set the style and alternative of the used font. The parameter mechanism is rather flexible so that within the parameter