The font name mess

Keywords
ConTeXt mkiv, luatex, font names

Introduction

When TEX came around it shipped with its own fonts. At that moment the TEX font universe was a small and well known territory. The 'only' hassle was that one needed to make sure that the right kind of bitmap was available for the printer.

When other languages than English came into the picture things became more complex as now fonts instances in specific encodings showed up. After a couple of years the by then standardised TEX distributions carried tens of thousands of font files. The reason for this was simple: TEX fonts could only have 256 characters and therefore there were quite some encodings. Also, large cjk fonts could easily have hundreds of metric files per font. Distributions also provide metrics for commercial fonts although I could never use them and as a result have many extra metric files in my personal trees (generated by TEXfont). (Distributions like TEXLive have between 50.000 and 100.000 files, but derivatives like the ConTEXt minimals are much smaller.)

At the input side many problems related to encodings were solved by Unicode. So, when the more Unicode aware fonts showed up, it looked like things would become easier. For instance, no longer were choices for encodings needed. Instead one had to choose features and enable languages and scripts and so the problem of the multitude of files was replaced by the necessity to know what some font actually provides. But still, for the average user it can be seen as an improvement.

A rather persistent problem remained, especially for those who want to use different fonts and or need to install fonts on the system that come from elsewhere (either free or commercial): the names used for fonts. You may argue that modern TEX engines and macro packages can make things easier, especially as one can call up fonts by their names instead of their file names, but actually the problem has worsened. With traditional TEX you definitely get an error when you mistype a file name or call for a font that is not on your system. The more modern TEX's macro packages can provide fall-back mechanisms and you can end up with something you didn't ask for.

For years one of the good things of TEX was its stability. If we forget about changes in content, macro packages and/or hyphenation patterns, documents could render more or less the same for years. This is because fonts didn't change. However, now that fonts are more complex, bugs gets fixed and thereby results can differ. Or, if you use platform fonts, your updated operating system might have new or even different variants. Or, if you access your fonts by font name, a lookup can resolve differently.

The main reason for this is that font names as well as file names of fonts are highly inconsistent across vendors, within vendors and platforms. As we have to deal with this matter, in MkIV we have several ways to address a font: by file name, by font name, and by specification. In the next sections I will describe all three.

Method 1: file

The most robust way to specify what fonts is to be used is the file name. This is done as follows:

```
\definefont[SomeFont][file:lmmono10-regular]
```

A file name lookup is case insensitive and the name you pass is exact. Of course the `file:` prefix (as with any prefix) can be used in font synonyms as well. You may add a suffix, so this is also valid:

```
\definefont[SomeFont][file:lmmono10-regular.otf]
```

By default ConTEXt will first look for an OpenType font so in both cases you will get such a font. But how do you know what the file name is? You can for instance check it out with:

```
mtxrun --script font --list --method=file --pattern="lm*mono" --all
```

This reports some information about the file, like the weight, style, width, font name, file name and optionally the subfont id and a mismatch between the analysed weight and the one mentioned by the font.

```
latinmodernmonolight      light   normal  normal     lmmonolt10regular          lmmonolt10-regular.otf
latinmodernmonoproplight  light   italic  normal     lmmonoproplt10oblique      lmmonoproplt10-oblique.otf
latinmodernmono           normal  normal  normal     lmmono9regular             lmmono9-regular.otf
latinmodernmonoprop       normal  italic  normal     lmmonoprop10oblique        lmmonoprop10-oblique.otf
latinmodernmono           normal  italic  normal     lmmono10italic             lmmono10-italic.otf
latinmodernmono           normal  normal  normal     lmmono8regular             lmmono8-regular.otf
latinmodernmonolightcond  light   italic  condensed  lmmonoltcond10oblique      lmmonoltcond10-oblique.otf
latinmodernmonolight      light   italic  normal     lmmonolt10oblique          lmmonolt10-oblique.otf
latinmodernmonolightcond  light   normal  condensed  lmmonoltcond10regular      lmmonoltcond10-regular.otf
latinmodernmonolight      bold    italic  normal     lmmonolt10boldoblique      lmmonolt10-boldoblique.otf
latinmodernmonocaps       normal  italic  normal     lmmonocaps10oblique        lmmonocaps10-oblique.otf
latinmodernmonoproplight  bold    italic  normal     lmmonoproplt10boldoblique  lmmonoproplt10-boldoblique.otf
latinmodernmonolight      bold    normal  normal     lmmonolt10bold             lmmonolt10-bold.otf
latinmodernmonoproplight  bold    normal  normal     lmmonoproplt10bold         lmmonoproplt10-bold.otf
latinmodernmonoslanted    normal  normal  normal     lmmonoslant10regular       lmmonoslant10-regular.otf
latinmodernmono           normal  normal  normal     lmmono12regular            lmmono12-regular.otf
latinmodernmonocaps       normal  normal  normal     lmmonocaps10regular        lmmonocaps10-regular.otf
latinmodernmonoprop       normal  normal  normal     lmmonoprop10regular        lmmonoprop10-regular.otf
latinmodernmono           normal  normal  normal     lmmono10regular            lmmono10-regular.otf
latinmodernmonoproplight  light   normal  normal     lmmonoproplt10regular      lmmonoproplt10-regular.otf
```

Method 2: name

Instead of lookup by file, you can also use names. In the font database we store references to the font name and full name as well as some composed names from information that comes with the font. This permits rather liberal naming and the main reason is that we can more easily look up fonts. In practice you will use names that are as close to the file name as possible.

```
mtxrun --script font --list --method=name --pattern="lmmono*regular" --all
```

This gives on my machine:

```
lmmono10regular         lmmono10regular         lmmono10-regular.otf
lmmono12regular         lmmono12regular         lmmono12-regular.otf
lmmono8regular          lmmono8regular          lmmono8-regular.otf
lmmono9regular          lmmono9regular          lmmono9-regular.otf
lmmonocaps10regular     lmmonocaps10regular     lmmonocaps10-regular.otf
lmmonolt10regular       lmmonolt10regular       lmmonolt10-regular.otf
```

```
lmmonoltcond10regular   lmmonoltcond10regular   lmmonoltcond10-regular.otf
lmmonoprop10regular     lmmonoprop10regular     lmmonoprop10-regular.otf
lmmonoproplt10regular   lmmonoproplt10regular   lmmonoproplt10-regular.otf
lmmonoslant10regular    lmmonoslant10regular    lmmonoslant10-regular.otf
```

It does not show from this list but with name lookups first OpenType fonts are checked and then Type1. In this case there are Type1 variants as well but they are ignored. Fonts are registered under all names that make sense and can be derived from their description. So:

```
mtxrun --script font --list --method=name --pattern="latinmodern*mono" --all
```

will give:

```
latinmodernmono                     lmmono9regular              lmmono9-regular.otf
latinmodernmonocaps                 lmmonocaps10oblique         lmmonocaps10-oblique.otf
latinmodernmonocapsitalic           lmmonocaps10oblique         lmmonocaps10-oblique.otf
latinmodernmonocapsnormal           lmmonocaps10oblique         lmmonocaps10-oblique.otf
latinmodernmonolight                lmmonolt10regular           lmmonolt10-regular.otf
latinmodernmonolightbold            lmmonolt10boldoblique       lmmonolt10-boldoblique.otf
latinmodernmonolightbolditalic      lmmonolt10boldoblique       lmmonolt10-boldoblique.otf
latinmodernmonolightcond            lmmonoltcond10oblique       lmmonoltcond10-oblique.otf
latinmodernmonolightconditalic      lmmonoltcond10oblique       lmmonoltcond10-oblique.otf
latinmodernmonolightcondlight       lmmonoltcond10oblique       lmmonoltcond10-oblique.otf
latinmodernmonolightitalic          lmmonolt10oblique           lmmonolt10-oblique.otf
latinmodernmonolightlight           lmmonolt10regular           lmmonolt10-regular.otf
latinmodernmononormal               lmmono9regular              lmmono9-regular.otf
latinmodernmonoprop                 lmmonoprop10oblique         lmmonoprop10-oblique.otf
latinmodernmonopropitalic           lmmonoprop10oblique         lmmonoprop10-oblique.otf
latinmodernmonoproplight            lmmonoproplt10oblique       lmmonoproplt10-oblique.otf
latinmodernmonoproplightbold        lmmonoproplt10boldoblique   lmmonoproplt10-boldoblique.otf
latinmodernmonoproplightbolditalic  lmmonoproplt10boldoblique   lmmonoproplt10-boldoblique.otf
latinmodernmonoproplightitalic      lmmonoproplt10oblique       lmmonoproplt10-oblique.otf
latinmodernmonoproplightlight       lmmonoproplt10oblique       lmmonoproplt10-oblique.otf
latinmodernmonopropnormal           lmmonoprop10oblique         lmmonoprop10-oblique.otf
latinmodernmonoslanted              lmmonoslant10regular        lmmonoslant10-regular.otf
latinmodernmonoslantednormal        lmmonoslant10regular        lmmonoslant10-regular.otf
```

Watch the 9 point version in this list. It happens that there are 9, 10 and 12 point regular variants but all those extras come in 10 point only. So we get a mix and if you want a specific design size you really have to be more specific. Because one font can be registered with its font name, full name etc. it can show up more than once in the list. You get what you ask for.

With this obscurity you might wonder why names make sense as lookups. One advantage is that you can forget about special characters. Also, Latin Modern with its design sizes is probably the worst case. So, although for most fonts a name like the following will work, for Latin Modern it gives one of the design sizes:

```
\definefont[SomeFont][name:latinmodernmonolightbolditalic]
```

But this is quite okay:

```
\definefont[SomeFont][name:lmmonolt10boldoblique]
```

So, in practice this method will work out as well as the file method but you can best check if you get what you want.

Method 3: spec

We have now arrived at the third method, selecting by means of a specification. This time we take the family name as starting point (although we have some fall-back mechanisms):

```
\definefont[SomeSerif]          [spec:times]
\definefont[SomeSerifBold]      [spec:times-bold]
\definefont[SomeSerifItalic]    [spec:times-italic]
\definefont[SomeSerifBoldItalic][spec:times-bold-italic]
```

The patterns are of the form:

```
spec:name-weight-style-width
spec:name-weight-style
spec:name-style
```

When only the name is used, it actually boils down to:

```
spec:name-normal-normal-normal
```

So, this is also valid:

```
spec:name-normal-italic-normal
spec:name-normal-normal-condensed
```

Again we can consult the database:

```
mtxrun --script font --list --method=spec lmmono-normal-italic --all
```

This prints the following list. The first column is the family name, the fifth column the font name:

```
latinmodernmono       normal   italic   normal   lmmono10italic        lmmono10-italic.otf
latinmodernmonoprop   normal   italic   normal   lmmonoprop10oblique   lmmonoprop10-oblique.otf
lmmono10              normal   italic   normal   lmmono10italic        lmtti10.afm
lmmonoprop10          normal   italic   normal   lmmonoprop10oblique   lmvtto10.afm
lmmonocaps10          normal   italic   normal   lmmonocaps10oblique   lmtcso10.afm
latinmodernmonocaps   normal   italic   normal   lmmonocaps10oblique   lmmonocaps10-oblique.otf
```

Watch the OpenType and Type1 mix. As we're just investigating here, the lookup looks at the font name and not at the family name. At the TEX end you use the family name:

```
\definefont[SomeFont][spec:latinmodernmono-normal-italic-normal]
```

So, we have the following ways to access this font:

```
\definefont[SomeFont][file:lmmono10-italic]
\definefont[SomeFont][file:lmmono10-italic.otf]
\definefont[SomeFont][name:lmmono10italic]
\definefont[SomeFont][spec:latinmodernmono-normal-italic-normal]
```

As OpenType fonts are preferred over Type1 there is not much chance of a mix-up.

As mentioned in the introduction, qualifications are somewhat inconsistent. Among the weight we find: black, bol, bold, demi, demibold, extrabold, heavy, light, medium, mediumbold, regular, semi, semibold, ultra, ultrabold and ultralight. Styles are: ita, ital, italic, roman, regular, reverseoblique, oblique and slanted. Examples of width are: book, cond, condensed, expanded, normal and thin. Finally we have alternatives which can be anything.

When doing a lookup, some normalizations takes place, with the default always being 'normal'. But still the repertoire is large:

```
helveticaneue medium     normal normal    helveticaneuemedium           HelveticaNeue.ttc index:  0
helveticaneue bold       normal condensed helveticaneuecondensedbold    HelveticaNeue.ttc index:  1
helveticaneue black      normal condensed helveticaneuecondensedblack   HelveticaNeue.ttc index:  2
helveticaneue ultralight italic thin      helveticaneueultralightitalic HelveticaNeue.ttc index:  3
helveticaneue ultralight normal thin      helveticaneueultralight       HelveticaNeue.ttc index:  4
helveticaneue light      italic normal    helveticaneuelightitalic      HelveticaNeue.ttc index:  5
helveticaneue light      normal normal    helveticaneuelight            HelveticaNeue.ttc index:  6
helveticaneue bold       italic normal    helveticaneuebolditalic       HelveticaNeue.ttc index:  7
helveticaneue normal     italic normal    helveticaneueitalic           HelveticaNeue.ttc index:  8
helveticaneue bold       normal normal    helveticaneuebold             HelveticaNeue.ttc index:  9
helveticaneue normal     normal normal    helveticaneue                 HelveticaNeue.ttc index: 10
helveticaneue normal     normal condensed helveticaneuecondensed        hlc_____.afm   conflict: roman
helveticaneue bold       normal condensed helveticaneueboldcond         hlbc____.afm
helveticaneue black      normal normal    helveticaneueblackcond        hlzc____.afm   conflict: normal
helveticaneue black      normal normal    helveticaneueblack            hlbl____.afm   conflict: normal
helveticaneue normal     normal normal    helveticaneueroman            lt_50259.afm   conflict: regular
```

The font database

In MkIV we use a rather extensive font database which in addition to bare information also contains a couple of hashes. When you use ConTEXt MkIV and install a new font, you have to regenerate the file database. In a next TEX run this will trigger a reload of the font database. Of course you can also force a reload with:

```
mtxrun --script font --reload
```

As a summary we mention a few of the discussed calls of this script:

```
mtxrun --script font --list somename (== --pattern=*somename*)

mtxrun --script font --list --method=name somename
mtxrun --script font --list --method=name --pattern=*somename*

mtxrun --script font --list --method=spec somename
mtxrun --script font --list --method=spec somename-bold-italic
mtxrun --script font --list --method=spec --pattern=*somename*
mtxrun --script font --list --method=spec --filter="fontname=somename"
mtxrun --script font --list --method=spec
                     --filter="familyname=somename,weight=bold,style=italic,width=condensed"

mtxrun --script font --list --method=file somename
mtxrun --script font --list --method=file --pattern=*somename*
```

The lists shown in before depend on what fonts are installed and their version. They might not reflect reality at the time you read this.

Interfacing

Regular users never deal with the font database directly. However, if you write font loading macros yourself, you can access the database from the TEX end. First we show an example of an entry in the database, in this case TeXGyreTermes Regular.

```
{
    designsize = 100,
    familyname = "texgyretermes",
    filename = "texgyretermes-regular.otf",
    fontname = "texgyretermesregular",
    fontweight = "regular",
    format = "otf",
    fullname = "texgyretermesregular",
    maxsize = 200,
    minsize = 50,
    rawname = "TeXGyreTermes-Regular",
    style = "normal",
    variant = "",
    weight = "normal",
    width = "normal",
}
```

Another example is Helvetica Neue Italic:

```
{
    designsize = 0,
    familyname = "helveticaneue",
    filename = "HelveticaNeue.ttc",
    fontname = "helveticaneueitalic",
    fontweight = "book",
    format = "ttc",
    fullname = "helveticaneueitalic",
    maxsize = 0,
    minsize = 0,
    rawname = "Helvetica Neue Italic",
    style = "italic",
    subfont = 8,
    variant = "",
    weight = "normal",
    width = "normal",
}
```

As you can see, some fields can be meaningless, like the sizes. As using the low level TEX interface assumes some knowledge, we stick here to an example:

```
\def\TestLookup#1%
  {\dolookupfontbyspec{#1}
   pattern: #1, found: \dolookupnoffound
   \blank
   \dorecurse {\dolookupnoffound} {%
     \recurselevel:~\dolookupgetkeyofindex{fontname}{\recurselevel}%
```

```
    \quad
  }%
  \blank}

\TestLookup{familyname=helveticaneue}
\TestLookup{familyname=helveticaneue,weight=bold}
\TestLookup{familyname=helveticaneue,weight=bold,style=italic}
```

You can use the following commands:

```
\dolookupfontbyspec    {key=value list}
\dolookupnoffound
\dolookupgetkeyofindex {key}{index}
\dolookupgetkey        {key}
```

First you do a lookup. After that there can be one or more matches and you can access the fields of each match. What you do with the information is up to yourself.

A few remarks

The fact that modern TEX engines can access system fonts is promoted as a virtue. The previous sections demonstrated that in practice this does not really free us from a name mess. Of course, when we use a really small TEX tree, and system fonts only, there is not much that can go wrong, but when you have extra fonts installed there can be clashes.

We're better off with file names than we were in former times when operating systems and media forced distributors to stick to 8 characters in file names. But that does not guarantee that today's shipments are more consistent. And as there are still some limitations in the length of font names, obscure names will be with us for a long time to come.

Hans Hagen