Font installation example: IBM Plex

Taco Hoekwater

Installing and using a new font family for use with ConTEXt is not all that hard, but it can be a bit daunting for an inexperienced user. This article shows an example using the free font family 'IBM Plex'.

1. Installation

Actual installation of a font is by far the simplest part of this process.

If you want to use the font with ConTEXt exclusively, just copy all the font files (otf or ttf) into the `fonts/data` directory below `texmf-fonts` in your installation. Then update the ConTEXt filename database:

```
> mtxrun --generate
```

If you also want to use the font family in your operating system, then use the normal procedure for installing new fonts for your system. You could skip the previous step in this case since system fonts are always found independent of the filename database, but I find it useful to have all my ConTEXt fonts somewhere within the distribution. It means they can easily be found in backups, et cetera. So, I tend to do both.

Regardless, next you probably want to update the ConTEXt font name database:

```
> mtxrun --script fonts --reload
```

This can take a while, depending on how many fonts and font directories you have, but it should complete eventually without any errors. If you do get errors in this stage, then you should ask for advice on the mailing list, as it will be something odd that cannot be predicted by me.

At this point it makes sense to make sure that everything is ok. That means asking `mtxrun` to produce the list of fonts, and checking that your new font file(s) are actually mentioned in its output. You could run

```
> mtxrun --script fonts --list --all
```

but the output from that can be huge. If you know (part of) the name of your font family, it is better to limit the output to just the fonts matching that name. In our example case, I am sure that the string `Plex` will appear in all font entries (as it is part of the file names), and also I know that I do not have a whole host of other fonts that would match that string. So:

```
> mtxrun --script fonts --list --all --pattern=Plex
```

```
resolvers       | trees | analyzing 'home:texmf'

identifier                          familyname            fontname                          filename                                    subfont   instances

ibmplexmono                         ibmplexmono           ibmplexmono                       IBMPlexMono-Regular.otf
ibmplexmonobold                     ibmplexmono           ibmplexmonobold                   IBMPlexMono-Bold.otf
ibmplexmonobolditalic               ibmplexmono           ibmplexmonobolditalic             IBMPlexMono-BoldItalic.otf
ibmplexmonoextralight               ibmplexmono           ibmplexmonoextralight             IBMPlexMono-ExtraLight.otf
ibmplexmonoextralightitalic         ibmplexmono           ibmplexmonoextralightitalic       IBMPlexMono-ExtraLightItalic.otf
ibmplexmonoitalic                   ibmplexmono           ibmplexmonoitalic                 IBMPlexMono-Italic.otf
ibmplexmonolight                    ibmplexmono           ibmplexmonolight                  IBMPlexMono-Light.otf
ibmplexmonolightitalic              ibmplexmono           ibmplexmonolightitalic            IBMPlexMono-LightItalic.otf
ibmplexmonomedium                   ibmplexmono           ibmplexmonomedium                 IBMPlexMono-Medium.otf
ibmplexmonomediumitalic             ibmplexmono           ibmplexmonomediumitalic           IBMPlexMono-MediumItalic.otf
ibmplexmononormal                   ibmplexmono           ibmplexmonoitalic                 IBMPlexMono-Italic.otf
ibmplexmonoregular                  ibmplexmono           ibmplexmonoextralight             IBMPlexMono-ExtraLight.otf
ibmplexmonosemibold                 ibmplexmono           ibmplexmonosemibold               IBMPlexMono-SemiBold.otf
ibmplexmonosemibolditalic           ibmplexmono           ibmplexmonosemibolditalic         IBMPlexMono-SemiBoldItalic.otf
ibmplexmonotext                     ibmplexmono           ibmplexmonotext                   IBMPlexMono-Text.otf
ibmplexmonotextitalic               ibmplexmono           ibmplexmonotextitalic             IBMPlexMono-TextItalic.otf
ibmplexmonothin                     ibmplexmono           ibmplexmonothin                   IBMPlexMono-Thin.otf
ibmplexmonothinitalic               ibmplexmono           ibmplexmonothinitalic             IBMPlexMono-ThinItalic.otf
ibmplexsans                         ibmplexsans           ibmplexsans                       IBMPlexSans-Regular.otf
ibmplexsansbold                     ibmplexsans           ibmplexsansbold                   IBMPlexSans-Bold.otf
ibmplexsansbolditalic               ibmplexsans           ibmplexsansbolditalic             IBMPlexSans-BoldItalic.otf
ibmplexsanscond                     ibmplexsanscondensed  ibmplexsanscond                   IBMPlexSansCondensed-Regular.otf
ibmplexsanscondbold                 ibmplexsanscondensed  ibmplexsanscondbold               IBMPlexSansCondensed-Bold.otf
ibmplexsanscondbolditalic           ibmplexsanscondensed  ibmplexsanscondbolditalic         IBMPlexSansCondensed-BoldItalic.otf
ibmplexsanscondensed                ibmplexsanscondensed  ibmplexsanscond                   IBMPlexSansCondensed-Regular.otf
ibmplexsanscondensedbold            ibmplexsanscondensed  ibmplexsanscondbold               IBMPlexSansCondensed-Bold.otf
ibmplexsanscondensedbolditalic      ibmplexsanscondensed  ibmplexsanscondbolditalic         IBMPlexSansCondensed-BoldItalic.otf
ibmplexsanscondensedextralight      ibmplexsanscondensed  ibmplexsanscondextralight         IBMPlexSansCondensed-ExtraLight.otf
ibmplexsanscondensedextralightitalic  ibmplexsanscondensed  ibmplexsanscondextralightitalic  IBMPlexSansCondensed-ExtraLightItalic.otf
ibmplexsanscondenseditalic          ibmplexsanscondensed  ibmplexsansconditalic             IBMPlexSansCondensed-Italic.otf
ibmplexsanscondensedlight           ibmplexsanscondensed  ibmplexsanscondlight              IBMPlexSansCondensed-Light.otf
ibmplexsanscondensedlightitalic     ibmplexsanscondensed  ibmplexsanscondlightitalic        IBMPlexSansCondensed-LightItalic.otf
ibmplexsanscondensedmedium          ibmplexsanscondensed  ibmplexsanscondmedium             IBMPlexSansCondensed-Medium.otf
ibmplexsanscondensedmediumitalic    ibmplexsanscondensed  ibmplexsanscondmediumitalic       IBMPlexSansCondensed-MediumItalic.otf
ibmplexsanscondensednormal          ibmplexsanscondensed  ibmplexsansconditalic             IBMPlexSansCondensed-Italic.otf
ibmplexsanscondensedregular         ibmplexsanscondensed  ibmplexsanscondextralight         IBMPlexSansCondensed-ExtraLight.otf
ibmplexsanscondensedsemibold        ibmplexsanscondensed  ibmplexsanscondsemibold           IBMPlexSansCondensed-SemiBold.otf
ibmplexsanscondensedsemibolditalic  ibmplexsanscondensed  ibmplexsanscondsemibolditalic     IBMPlexSansCondensed-SemiBoldItalic.otf
ibmplexsanscondensedtext            ibmplexsanscondensed  ibmplexsanscondtext               IBMPlexSansCondensed-Text.otf
ibmplexsanscondensedtextitalic      ibmplexsanscondensed  ibmplexsanscondtextitalic         IBMPlexSansCondensed-TextItalic.otf
ibmplexsanscondensedthin            ibmplexsanscondensed  ibmplexsanscondthin               IBMPlexSansCondensed-Thin.otf
ibmplexsanscondensedthinitalic      ibmplexsanscondensed  ibmplexsanscondthinitalic         IBMPlexSansCondensed-ThinItalic.otf
ibmplexsanscondextralight           ibmplexsanscondensed  ibmplexsanscondextralight         IBMPlexSansCondensed-ExtraLight.otf
ibmplexsanscondextralightitalic     ibmplexsanscondensed  ibmplexsanscondextralightitalic   IBMPlexSansCondensed-ExtraLightItalic.otf
ibmplexsansconditalic               ibmplexsanscondensed  ibmplexsansconditalic             IBMPlexSansCondensed-Italic.otf
ibmplexsanscondlight                ibmplexsanscondensed  ibmplexsanscondlight              IBMPlexSansCondensed-Light.otf
ibmplexsanscondlightitalic          ibmplexsanscondensed  ibmplexsanscondlightitalic        IBMPlexSansCondensed-LightItalic.otf
ibmplexsanscondmedium               ibmplexsanscondensed  ibmplexsanscondmedium             IBMPlexSansCondensed-Medium.otf
ibmplexsanscondmediumitalic         ibmplexsanscondensed  ibmplexsanscondmediumitalic       IBMPlexSansCondensed-MediumItalic.otf
ibmplexsanscondsemibold             ibmplexsanscondensed  ibmplexsanscondsemibold           IBMPlexSansCondensed-SemiBold.otf
ibmplexsanscondsemibolditalic       ibmplexsanscondensed  ibmplexsanscondsemibolditalic     IBMPlexSansCondensed-SemiBoldItalic.otf
ibmplexsanscondtext                 ibmplexsanscondensed  ibmplexsanscondtext               IBMPlexSansCondensed-Text.otf
ibmplexsanscondtextitalic           ibmplexsanscondensed  ibmplexsanscondtextitalic         IBMPlexSansCondensed-TextItalic.otf
ibmplexsanscondthin                 ibmplexsanscondensed  ibmplexsanscondthin               IBMPlexSansCondensed-Thin.otf
ibmplexsanscondthinitalic           ibmplexsanscondensed  ibmplexsanscondthinitalic         IBMPlexSansCondensed-ThinItalic.otf
ibmplexsansextralight               ibmplexsans           ibmplexsansextralight             IBMPlexSans-ExtraLight.otf
ibmplexsansextralightitalic         ibmplexsans           ibmplexsansextralightitalic       IBMPlexSans-ExtraLightItalic.otf
ibmplexsanshebrew                   ibmplexsanshebrew     ibmplexsanshebrew                 IBMPlexSansHebrew-Regular.otf
ibmplexsanshebrewbold               ibmplexsanshebrew     ibmplexsanshebrewbold             IBMPlexSansHebrew-Bold.otf
ibmplexsanshebrewextralight         ibmplexsanshebrew     ibmplexsanshebrewextralight       IBMPlexSansHebrew-ExtraLight.otf
ibmplexsanshebrewlight              ibmplexsanshebrew     ibmplexsanshebrewlight            IBMPlexSansHebrew-Light.otf
ibmplexsanshebrewmedium             ibmplexsanshebrew     ibmplexsanshebrewmedium           IBMPlexSansHebrew-Medium.otf
ibmplexsanshebrewnormal             ibmplexsanshebrew     ibmplexsanshebrew                 IBMPlexSansHebrew-Regular.otf
ibmplexsanshebrewregular            ibmplexsanshebrew     ibmplexsanshebrewextralight       IBMPlexSansHebrew-ExtraLight.otf
ibmplexsanshebrewsemibold           ibmplexsanshebrew     ibmplexsanshebrewsemibold         IBMPlexSansHebrew-SemiBold.otf
ibmplexsanshebrewtext               ibmplexsanshebrew     ibmplexsanshebrewtext             IBMPlexSansHebrew-Text.otf
ibmplexsanshebrewthin               ibmplexsanshebrew     ibmplexsanshebrewthin             IBMPlexSansHebrew-Thin.otf
ibmplexsansitalic                   ibmplexsans           ibmplexsansitalic                 IBMPlexSans-Italic.otf
ibmplexsanslight                    ibmplexsans           ibmplexsanslight                  IBMPlexSans-Light.otf
ibmplexsanslightitalic              ibmplexsans           ibmplexsanslightitalic            IBMPlexSans-LightItalic.otf
ibmplexsansmedium                   ibmplexsans           ibmplexsansmedium                 IBMPlexSans-Medium.otf
ibmplexsansmediumitalic             ibmplexsans           ibmplexsansmediumitalic           IBMPlexSans-MediumItalic.otf
ibmplexsansnormal                   ibmplexsans           ibmplexsansitalic                 IBMPlexSans-Italic.otf
ibmplexsansregular                  ibmplexsans           ibmplexsansextralight             IBMPlexSans-ExtraLight.otf
ibmplexsanssemibold                 ibmplexsans           ibmplexsanssemibold               IBMPlexSans-SemiBold.otf
ibmplexsanssemibolditalic           ibmplexsans           ibmplexsanssemibolditalic         IBMPlexSans-SemiBoldItalic.otf
ibmplexsanstext                     ibmplexsans           ibmplexsanstext                   IBMPlexSans-Text.otf
ibmplexsanstextitalic               ibmplexsans           ibmplexsanstextitalic             IBMPlexSans-TextItalic.otf
ibmplexsansthin                     ibmplexsans           ibmplexsansthin                   IBMPlexSans-Thin.otf
ibmplexsansthinitalic               ibmplexsans           ibmplexsansthinitalic             IBMPlexSans-ThinItalic.otf
ibmplexserif                        ibmplexserif          ibmplexserif                      IBMPlexSerif-Regular.otf
ibmplexserifbold                    ibmplexserif          ibmplexserifbold                  IBMPlexSerif-Bold.otf
ibmplexserifbolditalic              ibmplexserif          ibmplexserifbolditalic            IBMPlexSerif-BoldItalic.otf
ibmplexserifextralight              ibmplexserif          ibmplexserifextralight            IBMPlexSerif-ExtraLight.otf
ibmplexserifextralightitalic        ibmplexserif          ibmplexserifextralightitalic      IBMPlexSerif-ExtraLightItalic.otf
ibmplexserifitalic                  ibmplexserif          ibmplexserifitalic                IBMPlexSerif-Italic.otf
ibmplexseriflight                   ibmplexserif          ibmplexseriflight                 IBMPlexSerif-Light.otf
ibmplexseriflightitalic             ibmplexserif          ibmplexseriflightitalic           IBMPlexSerif-LightItalic.otf
ibmplexserifmedium                  ibmplexserif          ibmplexserifmedium                IBMPlexSerif-Medium.otf
ibmplexserifmediumitalic            ibmplexserif          ibmplexserifmediumitalic          IBMPlexSerif-MediumItalic.otf
ibmplexserifnormal                  ibmplexserif          ibmplexserifitalic                IBMPlexSerif-Italic.otf
ibmplexserifregular                 ibmplexserif          ibmplexserifextralight            IBMPlexSerif-ExtraLight.otf
ibmplexserifsemibold                ibmplexserif          ibmplexserifsemibold              IBMPlexSerif-SemiBold.otf
ibmplexserifsemibolditalic          ibmplexserif          ibmplexserifsemibolditalic        IBMPlexSerif-SemiBoldItalic.otf
ibmplexseriftext                    ibmplexserif          ibmplexseriftext                  IBMPlexSerif-Text.otf
ibmplexseriftextitalic              ibmplexserif          ibmplexseriftextitalic            IBMPlexSerif-TextItalic.otf
ibmplexserifthin                    ibmplexserif          ibmplexserifthin                  IBMPlexSerif-Thin.otf
ibmplexserifthinitalic              ibmplexserif          ibmplexserifthinitalic            IBMPlexSerif-ThinItalic.otf
```

Figure 1. Output of `mtxrun --script fonts --list --all --pattern=Plex`

Do not forget to add `--all` in this last command line, otherwise you will only get a single result back, instead of lines for all the fonts that match the pattern. In figure 1 you can see the output I get (in a teeny weeny monospaced font, otherwise it will not fit the article paper). Make sure all the font files you have installed appear somewhere in the output. If not, there is probably something odd about that font or its file permissions. Again, it is probably best to ask the mailing list for advice.

2. Understanding the `mtxrun` output

The output from `mtxrun` is formatted in columns. The meanings of the columns is as follows:

1. `identifier`
   This is one of the internal identifiers used within ConTEXt, used if you specify a font using the `spec:` prefix. If you look closely at the output list, you can see that there can be more than one line per font file, with the only difference being the `identifier`. This is because `spec:` attempts to convert the rather arbitrary font names to some semblance of order. My experiences with using the `spec:` identifier are not so good, and it is mostly helpful for higher-level modules like `selectfont`.
2. `familyname`
   This is a sanitized version of the familyname entry within the separate font files. This is also mostly helpful for `selectfont`, and we can safely ignore it.
3. `fontname`
   This is the font identifier that is used with ConTEXt if you specify a font using the `name:` prefix. This is my preferred method to specify fonts. It does not always work, though and that is where the next column comes in handy.
4. `filename`
   This is the font identifier that is used with ConTEXt if you specify a font using the `file:` prefix. Useful mostly if you have different versions of fonts with identical names in different font files. I use this for fonts that are in my ConTEXt distribution but that are also installed in my operating system, just to make sure that I have the right one in case one of them changes after an update.
5. `subfont`
   For fonts stored in Truetype font collections (`.ttc` files), this gives the used font index within the collection. In ConTEXt, you access such fonts by appending `(<index>)` to the file name if you are using the `file:` prefix.
6. `instances`
   For variable fonts, this lists the known instances of the font.

Incidentally, the values returned in the first columns can also be used as arguments to the various sub-options for the `--list` command.

3. Testing a few font instances

At this point, I usually opt for a bit of paranoia. I suggest you create a small test file:

```
\startTEXpage
\definedfont[name:ibmplexmono] \input knuth
\stopTEXpage
```

Do a visual inspection of the generated page to make sure it matches what you expect, and also check the `loaded fonts` line from the run's output, just in case. The output line should look something like this:

```
mkiv lua stats  > loaded fonts: 3 files: ibmplexmono-regular.otf,
latinmodern-math.otf, lmroman12-regular.otf
```

Nowadays, it seems that ConTEXt always preloads some fonts that are then not actually used (It needs a math font to initialize some of the internals, and since no explicit math font is defined yet it loads `latinmodern-math.otf` by default. Probably something similar makes it load `lmroman12-regular.otf`) but the important one is `ibmplexmono-regular.otf`, of course.

If you want, you can test all the new fonts separately, but usually if any database entry is ok, they are all ok.

4. Font repertoire

When you get a new font, I assume you have checked that the included glyphs are what you need for your texts. But you did that test on the company website, and it makes sense to do an extra test inside ConTEXt, just in case. For a quick latin test, you can use `\showfont`, but it is better to get the full list.

```
\usemodule[fnt-10]
\starttext
\ShowCompleteFont{name:ibmplexmono}{20pt}{1}
\stoptext
```

This will produce a pdf that lists each and every glyph in the font in a table that contains the glyph's unicode slot, the visual, the glyph number in the font, the glyph's actual name (if present), the glyph's expected name according to the Adobe Glyph list (if known) and optionally a list of ConTEXt command mappings.

Again, you could test all fonts, but usually one per family is enough. It is not unusual that there are differences between families, even if fonts within a family generally all have the same repertoire. In this case for example, the `ibmplexsans` has Cyrillic and greek but no Hebrew (in fact the `ibmplexsans` is the only family with greek), and `ibmplexsanshebrew` has Hebrew but not Cyrillic.

5. Feature discovery

Since this is an OpenType font set, there are probably features that can be turned on and off. To get a list of the features within a font, you can ask `mtxrun`:

```
mtxrun --script fonts --list --info ibmplexsans
```

A small part of that output looks like this:

```
mtx-fonts       | gpos features:
mtx-fonts       |
mtx-fonts       | feature  script   languages
mtx-fonts       |
mtx-fonts       | kern     cyrl     dflt
mtx-fonts       |          dflt     dflt
mtx-fonts       |          grek     dflt
mtx-fonts       |          latn     dflt
mtx-fonts       | mark     cyrl     dflt
mtx-fonts       |          dflt     dflt
mtx-fonts       |          grek     dflt
mtx-fonts       |          latn     dflt
mtx-fonts       |
mtx-fonts       | gsub features:
mtx-fonts       |
mtx-fonts       | feature  script   languages
mtx-fonts       |
mtx-fonts       | aalt     cyrl     dflt
mtx-fonts       |          dflt     dflt
mtx-fonts       |          grek     dflt
mtx-fonts       |          latn     dflt
...
```

Which tells us that there are two positioning features (`gpos`): kerning (`kern`) and accent placement (`mark`). This is fairly standard, and both features are turned on by default in ConTEXt so there is no need pay special attention to these.

We also see here that this font defines four scripts: Cyrillic (`cyrl`), Default (`dflt`), Greek (`grek`) and Latin (`latn`). There could potentially be more scripts that these features do not apply to, but that is highly unlikely in this case. The fact that languages is `dflt` simply means that the feature is to be applied when the language is set to default. In this font, that is the case for all features including the substitution (`gsub`) features, but it is not unusual to see `gsub` features that only apply to specific combinations of script and language. The Hebrew fonts in this example family define an extra feature `locl` for the language `iwr` in script `hebr`.

Off-screen, I ran this command for a number of different fonts, and it seems that they all have roughly the same feature set in this case, even across families. There are a few differences: only the sans fonts have greek, so they are the only ones that define the `grek` script. Only the Hebrew fonts define the `hebr` script. And the monospaced fonts do not have either `kern` or `liga`.

`kern`	Kerning	`mark`	Mark positioning
`aalt`	Access all alternates	`ccmp`	Glyph composition/decomposition
`dnom`	Denominators	`frac`	Fractions
`liga`	Standard ligatures	`numr`	Numerators
`ordn`	Ordinals	`salt`	Stylistic alternates
`sinf`	Scientific inferiors	`ss01`	Stylistic set 1
`ss02`	Stylistic set 2	`ss03`	Stylistic set 3
`ss04`	Stylistic set 4	`ss05`	Stylistic set 5
`sups`	Superscript	`zero`	Slashed zero

What this does not tell you is exactly what glyphs are affected by what feature. I know of no easy way to get that information, but presumably this information can be attained from the font source. Either that, or you will have to experiment ...

6. Creating typescripts

With all of the preliminary work done, it is time to create an initial version of the typescripts. For this, I usually work in a single document, with the typescripts-to-be defined in the setup section followed by one or more `\usetypescript` lines, and the body of the document containing nothing but repetitions of

```
\showbodyfont[typescriptname,12pt]
```

A full example will follow later.

For now, we need to deal with something else. Normally you set up the typescripts by taking one of each of the font families, and connecting them to one of the `Sans`, `Serif`, or `Mono` typefaces. In a traditional four-font font family, this is a simple process. But ConTEXt's roots in traditional TEX are quite obvious in this area: while there is support for three font styles (roman, italic, and slanted), there is only support for two weights (normal and bold). Modern font families nowadays typically have only two styles, but up to nine weights! ConTEXt also has support for a separate 'smallcaps' font, which is hardly ever used these days, instead handling caps and small caps as an opentype feature.

The simplest way out of this (at least, unless ConTEXt is extended sometime in the future) is to divide your font family into as many four-font subfamilies as needed, and create separate typescripts for each.

Let's have a look at the `ibmplexmono` font family. It has eight weights (thin, extra-light, light, normal, text, medium, semibold, and bold), each with a roman and an italic style. Weight names are not always easy to interpret so sometimes matching a ‘normal’ with a ‘bold’ can be trial and error, but in this case the combinations are fairly clear. Four separate typescripts are needed and the resulting test file looks like this:

```
\starttypescript [mono] [ibmplex-thin]
 \definefontsynonym[Mono]           [name:ibmplexmonothin]
 \definefontsynonym[MonoItalic]     [name:ibmplexmonothinitalic]
 \definefontsynonym[MonoBold]       [name:ibmplexmonotext]
 \definefontsynonym[MonoBoldItalic] [name:ibmplexmonotextitalic]
\stoptypescript

\starttypescript [mono] [ibmplex-extralight]
 \definefontsynonym[Mono]           [name:ibmplexmonoextralight]
 \definefontsynonym[MonoItalic] [name:ibmplexmonoextralightitalic]
 \definefontsynonym[MonoBold]       [name:ibmplexmonomedium]
 \definefontsynonym[MonoBoldItalic] [name:ibmplexmonomediumitalic]
\stoptypescript

\starttypescript [mono] [ibmplex-light]
 \definefontsynonym[Mono]           [name:ibmplexmonolight]
 \definefontsynonym[MonoItalic]     [name:ibmplexmonolightitalic]
 \definefontsynonym[MonoBold]       [name:ibmplexmonosemibold]
 \definefontsynonym[MonoBoldItalic][name:ibmplexmonosemibolditalic]
\stoptypescript

\starttypescript [mono] [ibmplex]
 \definefontsynonym[Mono]           [name:ibmplexmono]
 \definefontsynonym[MonoItalic]     [name:ibmplexmonoitalic]
 \definefontsynonym[MonoBold]       [name:ibmplexmonobold]
 \definefontsynonym[MonoBoldItalic] [name:ibmplexmonobolditalic]
\stoptypescript

\starttypescript [ibmplex-thin,ibmplex-extralight,
                  ibmplex-light,ibmplex]
 \definetypeface [\typescriptone]
                    [tt] [mono] [\typescriptone] [default]
\stoptypescript

\usetypescript[ibmplex-thin,ibmplex-extralight,
               ibmplex-light,ibmplex]

\starttext
\showbodyfont[ibmplex-thin,12pt]
\showbodyfont[ibmplex-extralight,12pt]
\showbodyfont[ibmplex-light,12pt]
```

```
\showbodyfont[ibmplex,12pt]
\stoptext
```

The two arguments [mono] [ibmplex] etcetera in the separate blocks are used by the third and fourth argument of \definetypeface. For a font family with serifs, you would use [serif], for a sans-serif family [sans] as the first argument. The second argument is your choice, but it makes some sense to reuse the names later on for the actual typescripts as that prevents confusion and allows shortcuts like using \typescriptone.

After running this file, some of it should look good. Of course all the entries for the \rm and \ss lines are wrong, but that is to be expected, only the lines for \tt are really interesting for now.

In each of those, the \tt\tf, \tt\it, \tt\bf, \tt\bi and all of the entries for \tfXXX should all use the just installed font.

But the ones for the slanted and small caps commands are still wrong. This is where it makes sense to compensate for the TEX history of ConTEXt font support. In case someone (or an external macro) uses the \sl, \bs or \sc commands, it is better to intercept the wrong font.

Normally I solve this by adding three lines to each of the \definefontsynonym blocks:

```
  \definefontsynonym [MonoSlanted]     [MonoItalic]
  \definefontsynonym [MonoBoldSlanted][MonoBoldItalic]
  \definefontsynonym [MonoCaps]        [Mono]
```

This makes the \sc command fairly useless. If the font actually had a small caps feature, it would be possible to set that up, by using

```
  \definefontsynonym
    [MonoCaps]
    [name:ibmplexmono]
    [features=smallcaps]
```

but this font family does not, so that won't help. Nevertheless, it is still better to see a non-small capped IBM Plex font with the right weight, than a small capped Latin Modern Mono in the standard weight.

The next task would be to set up the \definefontsynonym blocks for the other families. This is very similar to the block for the monospaced family, except that it makes sense to add [features=default] to all of the \definefontsynonym lines. And of course the blocks need to start with [sans] or [serif]. Since all the families have fonts in eight styles, it makes sense to reuse the typescript names for at least the ibmplexserif and ibmplexsans.

After all the font definitions have been made, it is possible to fill in the list of typefaces some more. At least, we can complete the entries for `\rm` and `\ss`:

```
\starttypescript [ibmplex-thin,ibmplex-extralight,
                  ibmplex-light,ibmplex]
 \definetypeface [\typescriptone]
                   [rm] [serif] [\typescriptone] [default]
 \definetypeface [\typescriptone]
                   [ss] [sans]  [\typescriptone] [default]
 \definetypeface [\typescriptone]
                   [tt] [mono]  [\typescriptone] [default]
\stoptypescript
```

I will not add all the listings here, but the final typescript file will appear on the meeting website.

Generally, the user will want to have a `\definetypeface` for math as well, but finding a nice match is a bit tricky.

For `ibmplexsanscondensed` and `ibmplexsanshebrew` new typescript names should be invented, if you want those as separate typescripts. Since there is only a condensed sans-serif, you may not want that one, and instead use the condensed fonts only for special occasions where the size does not have to be variable, like in headers and footers.

For uses like that, it may be a good idea to add a number of global `\fontsynonyms` to the typescript file, like so:

```
\definefontsynonym
  [SansCondensed]
  [name:ibmplexsanscond]
  [features=default]
```

This is not required, but it makes for cleaner use of `\definefont`.

For the Hebrew fonts, in this case it is better to set them up as fallbacks only, as explained in the next section.

7. Fallbacks

Let's step back a bit and deal with the Hebrew fonts. We want to use the Hebrew font when typesetting Hebrew, but we do not want to loose the ability to typeset Cyrillic. A good way to do that is to define the Hebrew fonts as fallback for the normal sans-serif fonts.

In Unicode, the Hebrew block is roughly from `uni0590` to `uni05FF`. Not all of the code points are filled, and of the filled ones, the IBM Plex fonts may not cover all. But it is still better than nothing.

We need to define a few font callbacks, here is one:

```
\definefontfallback
   [SansHebrewFallback]
   [SansHebrew]
   [0x0590-0x05ff]
   [check=yes,force=no]
```

When this is used inside a third argument to a `\definefontsynonym` line, it tells ConTEXt that whenever the current font does not have a glyph in the Unicode range from hexadecimal 0590 up to 05ff, to use the substitute font `SansHebrew` instead. I used `force=no` here, because I expect somehow that the Hebrew block will eventually be merged into the normal sans font, and this offers some future proofing: with `force=yes`, any glyphs in the affected block in the base font would be permanently ignored.

We actually need two of those. The one from above, and also a version for `SansBoldHebrew`. I defined those globally, because they can be reused within the four different weight typescripts. We will let the italic styles fall back to upright, because there are no italic versions of the 'Sans Hebrew' font.

We need to use the `Hebrew` font feature for the Hebrew fonts instead of the `default`, which is one of the predefined font feature sets in ConTEXt.

Now we have to go back to the four typescript definitions for `sans`. We need to add the definition for the `SansHebrewXXXX` fonts, and adjust the `\definefontsynonym` lines. The end result of just one of the four blocks would look like this:

```
\starttypescript [sans] [ibmplex]
  \definefontsynonym
    [SansHebrew]
    [name:ibmplexsanshebrew] [features=hebrew]
  \definefontsynonym
    [SansHebrewBold]
    [name:ibmplexsanshebrewbold] [features=hebrew]

  \definefontsynonym
    [Sans]
    [name:ibmplexsans]
    [features=default, fallbacks=SansHebrewFallback]
  \definefontsynonym
    [SansItalic]
    [name:ibmplexsansitalic]
    [features=default, fallbacks=SansHebrewFallback]
  \definefontsynonym
    [SansBold]
    [name:ibmplexsansbold]
```

```
    [features=default, fallbacks=SansHebrewBoldFallback]
  \definefontsynonym
    [SansBoldItalic]
    [name:ibmplexsansbolditalic]
    [features=default, fallbacks=SansHebrewBoldFallback]

  \definefontsynonym [SansSlanted]    [SansItalic]
  \definefontsynonym [SansBoldSlanted][SansBoldItalic]
  \definefontsynonym [SansCaps]       [Sans]
\stoptypescript
```

Now we can write Hebrew as well as Cyrillic using only the simple `\ss` command. Do not forget to change the writing direction!

After filling in the other typescript blocks, we are finished with the typescript file!

8. Stylistic alternates and other features

So what about the other features like `ss01` and `zero`? The `zero` feature is a simple one to test, because per the definition in the OpenType standard it toggles on the use of a special kind of zero, like so:

```
\definedfont[Sans]
  {There is a 0 chance of \addff{zero} 0 changes.}
```

There is a 0 chance of 0 changes.

In this case, it was easiest to use the `\addff` command, but that will only work for some specific simple features. Quite often, special features need to use the Lua based OpenType processor that is set up with `mode=node` in a `\definefontfeature`.

The features that deal with stylistic sets are harder to discover. For this demonstration, I looked at the font using a font editor. It turns out that the first two are used to switch to an ‘italic’ style of certain characters. `ss01` does the various ‘a’- and ‘alpha’-based glyphs and `ss02` the ‘g’-based glyphs. Feature `ss03` is an alias for feature `zero`, features `ss04` produces a zero with a dot in the middle, and `ss05` changes the appearance of the ß glyph.

```
\definefontfeature
  [mynormal]
  [mode=node,ss01=yes,ss02=yes,ss04=yes,ss05=yes]

\definedfont[Sans*default]{There is a 0 chance of changes ß.}

\definedfont[Sans*mynormal]{There is a 0 chance of changes ß.}
```

There is a 0 chance of changes ß.

There is ɑ 0 chɑnce of chɑnges ß.

The `frac` feature is supposed to produce inline fractions from ascii input. To find out what the exact output is, it is best to run a small test:

```
\definefontfeature
  [mynormal]
  [mode=node,frac=yes,ss04=yes]

\definedfont[Sans*default]{1/2 1/3 2/3 1/4 2/4 3/4 5/9 20/42}

\definedfont[Sans*mynormal]{1/2 1/3 2/3 1/4 2/4 3/4 5/9 20/42}
```

1/2 1/3 2/3 1/4 2/4 3/4 5/9 20/42

$^{1}/_{2}$ $^{1}/_{3}$ $^{2}/_{3}$ $^{1}/_{4}$ $^{2}/_{4}$ $^{3}/_{4}$ $^{5}/_{9}$ $^{20}/_{42}$

As you can see, in this font the coverage is very good. This is because it replaces all glyphs in the numeric range, first all to their superscript variant, then when it sees the slash, it switches to the subscript variant. Some fonts implement the `frac` feature by converting the input into a ligature, and then coverage depends on how many ligatures are provided.

Depending on your font, sometimes features may depend on other features. The font documentation is supposed to help, but if it does not, then sometimes the only thing to do is to write a test file experimenting with turning all the features on and off and looking for differences in the output.

Afterword

This article came about because Pavneet Arora asked me for help on how to set up the IBM Plex family. Then we had the ConTEXt meeting in Sibřina, and the general consensus was that we preferred the IBM Plex font family over Alwyn New, the font family that was used thus far to typeset the Context group journal. The three big advantages of IBM Plex are: improved readability, a larger glyph repertoire, and it being a freely distributable font.

The Context group has not abandoned Alwyn New. We will keep using that family for all official communications. But the journal will from now on be typeset in IBM Plex Sans and Mono.