feature/all-tex-fonts

With a serendipity

Pedro A. Aranda Gutiérrez
mailto:paaguti@gmail.com

May 15, 2025

Motivation

- I picked up the discussion regarding font management
  - There had been a lot of traffic going around
  - Using `pdflatex` vs. `lualatex`, ...
- It was cumbersome to have a long `#+LATEX_HEADER:` preamble with all definitions, I had to repeat for each document.
- First steps (what you didn't see):
  - Learn more on `lualatex` and `fontspec`
  - Come up with font configurations (as `.tex`)
  - Use the charset detection code by Juan Manuel Macías
  - Generate font configurations using `elisp` to a `.tex` file.
- When I finally was able to create the feature branch:
  - Integrate code in the LaTeX generation logic.
  - No external TeX files needed

What's new?

- In the templates:
  - A new `[FONTSPEC]` keyword to place the `fontspec` configuration
- In the configuration
  - A new variable `org-latex-fontspec-config` to store font specifications
  - Can be stored in the Emacs initialisation code or in a file or directory local variable.
  - It was initially called `org-latex-lualatex-font-config`, which was cumbersome but reflected my concentrating on `lualatex`.

`org-latex-fontspec-config`

- It is driven by the `\set..font{}` commands needed in the `fontspec` configuration
- And by the charset definitions extracted with Juan Manuel's code
- Setting `org-latex-fontspec-config` to `nil` omits the `fontspec` configuration
  - In presentations, I generally must use the fonts defined in the template.
  - This is its current default value.

org-latex-fontspec-config

Fonts and basic features

- Fonts and their basic features are always generated[1]

```
((org-mode
  . ((org-latex-fontspec-config
      . (("main" :font "FreeSerif")
         ("sans" :font "Noto Sans")
         ("math" :font "Latin Modern Math")
         ("mono" :font "Noto Sans Mono"
                 :features "Scale=MatchLowercase"))))))
```

(1) this is a `.dir-locals.el`

- A definition for each and every font will be generated:

```
\setmainfont{FreeSerif}
\setsansfont{Noto Sans}
\setmainfont{Latin Modern Math}
\setmonofont{NotoSansMono}[Scale=MatchLowercase]
```


org-latex-fontspec-config

Fallback fonts

- The fallback features map script names with their fallback
- They will be generated only for scripts present in the document:

```
(("sans" :font "Noto Sans"
         :fallback (("emoji" . "Noto Color Emoji:mode=harf")
                    ("han"   . "Noto Sans CJK JP:")
                    ("kana"  . "Noto Sans CJK JP:"))))
```

- will generate

```
\directlua{
 luaotfload.add_fallback ("fallback_sans",{
  "Noto Color Emoji:mode=harf",
 })
\setsansfont{Noto Sans}[RawFeature={fallback=fallback_sans}]
```

On a document where the `emoji` script has been detected.

org-latex-fontspec-config

Fallback fonts: notes

- The `directlua` part will include all fallback definitions
- for scripts that are *present* in the document.
- It is placed before the `\set...font{}` statements.
- The `RawFeature` will be appended to other features a font might have.

Evolution

1. Test, test and test again (please)
   - Any complex setup for `lualatex` or `xelatex` that doesn't produce the expected result... please rework it on an MWE you can share.
2. Discussion: do we want this for `pdflatex`?
   - Complex font setups
   - Full working MWEs very much appreciated.
3. Detect and filter out commented text in charset detection?
4. When will the feature branch make to `main`?
   - Current state: Included 3 tests earlier today... ready for review???
5. `beamer`
   - Rearrange template:
     - Beamer specifics directly after `\documentclass{beamer}`
     - `fontspec` after that (if needed)
   - Specific `BEAMER_CLASS_ATTRIBUTES` ??
     - As a first step towards supporting `beamerarticle`