gropdf.zig Test

by

Sven Schober

Introduction and Motivation

This file contains only some paragraphs of text. Its purpose is to showcase some of groff's features und highlight potential rendering problems in my custom PDF backend, gropdf.zig.

Current State

Currently, a lot of features are implemented: We clearly can typeset characters, sentences and whole paragraphs. In addition, headings are working, some glitches have been solved by me. Even Drop Caps are working, and if you look closely, you will notice the page number at the bottom.

The positioning in grout is an interessting topic, as it seems to be absolute, at first glance. Meaning, there are `H` and `V` absolute coordinates given through all of the document.

But, when ligatures are to be rendered, `C` commands are issued and these do not increment the internal drawing position of the backend device. The same is true for wh commands, which increment the `x` drawing position in a relative way. But in order to be able to compute the new absolute position, we need to know the current position.

PDF on the other hand has a text rendering commands `Tj`, which increases the drawing position automatically, but we have no way to query the current position as that is an internal state of the renderer, which lives in the viewer.

There was no way around implementing relative positioning in `gropdf.zig`. That meant, we need to read the font description files residing in `/usr/share/groff/current/font/devpdf/<font name>`, where for every glyph its width is recorded. We use that when typsetting words with the `Tj` command to increase the internal `y` axis pointer.

That way positioning improved automatically.

Font Selection

Selecting the currently correct font is quiet involved, when bridging `groff_out(5)` and PDF. Groff issues commands like the following: `x font 38 TI` to mount a certain font at a certain position, but pdf needs a font to be registered on the document level and then again at the page level. The first registration yields an object number, the second a page relative font number. But from the driver, our `Transpiler` we are not interessted in the object number, but the index under which the font was registered in the document' s font list, which is not the same. We memorize both indices and use them during `fN`, a.ka. font select, operations.

Before registering a groff font with the document, we need to know how it is called in PDF. For that, we use a comptime zig map, called `StaticStringMap`, which is initialized and filled during compilation.

Character Mapping and Ligatures

Ligatures are special character combinations, like `'fi'`, or `'fl'`, that are commonly rendered as one special combined character. Typographically, this is because `'f'` and `'i'` for example can fit closer together than other character combinations.

PDF has a so called standard encoding, where these ligatures are assigned to certain code points. So, I used a zig compile time hash map to map the grout glyph names to these code points.

Drawing Commands

I just came around to implement drawing commands, `Dl` and `Dl`. Just enough to have the header rule render correctly. In PDF I just had to put some `m` and `l` and `S` commands and implementation wise I could heavily rely on what was already present in `TextObject`.

Based on that, I introduced a `GraphicalObject` that has very similar structure as `TextObject`, a line buffer and buffered lines, which can be rendered out to the PDF file at the end of processing.