MOTIVATION

In the past, I have written a little library, I call `mato` - short for Markdown Transformer framework[1]. In that library, I make heavy use of `groff` - GNU roff - which does the job of transforming output, my library generates, into PDF files. It does that fairly quickly, say around half a second for the average document.

But this step is by far the most expensive step in a whole chain of transformation steps: first `mato` reads the input file, and creates a abstract syntaxt tree from it. That tree is being processed several times and then rendered into `groff` input format. After that `groff` takes over. It is itself built in a similar way as a compiler, it has a front-end doing the parsing and a backend, doing the ouput generation. These backends are called *devices* in `groff` parlance.

One such device, I use heavily is the PDF device, implemented by `gropdf.pl`. As the name suggests, it is written in Perl[2] and thus rather expensive with regards to CPU consumption.

Because, I wanted to build something in `zig`, I decided, this would be a good toy project, to practise my skills.

FIRST ITERATION - GETTING SOMETHING

My first goal is to see some out put in macos' preview app. I have a sample PDF file copied over from a book called "PDF Explained" by O'Reilly. That book served as a starting

[1] I now notice for the first time, that the `o` character is out of place. It is a historical left-over, because initially, I called it `matote`, Markdown to TeX converter, but at some point dropped the TeX and switch it to groff, because it is much simpler and much quicker.

[2] Some of you might know this scripting language still!

point, but I soon learned, that the Adobe “PDF Reference 1.7” (see here).

The first problem my current implementation has, is that the output stream is chosen in the pdf implementation. So, the first challgenge becomes, factoring that out.