typho roadmap

Version 2023-01-29

Index

About this document

- This document is meant to specify next goals to streamline the efforts within the typho project.
- Simultaneously it serves as documentation for the public.
- “Goals” list steps to be achieve with this milestone in a high-level description. “Education” lists steps to improve my own understanding of the field or its expression to the general public. “Implementation” lists steps in terms of software deliverables. An optional list “Decisions” defines decisions I plan to use/assume for upcoming milestones.

Current status

- My current prototype implementation did not receive development for several months.
- I tested many ideas about hooks and content trees in the side project “litua”.
- I will get into touch with the public for the first time to collect some feedback.
- I am currently working on Milestone 0.2

Milestone 0.1

Goals

- Get a more comprehensive overview over the community.
- Gain some experience with first attempts implementing typesetting ideas in rust.

Education

- Investigate how to use the language server protocol for markup languages
- Document the generic data model used by pandoc for markup languages
- Look at the basic definitions of XML
- Share an online chat with two experts from Germany
- Look into value delegation implemented in Lua
- Study the requirements of syntax escaping
- Understand how xmltex work
- Systematically evaluate the set of crates in the field of text processing
- Investigate the intersection of LISP and "text documents as trees"
- Learn about Racket, Scribble, and "A System for Typesetting Mathematics" by B. Kernighan

Implementation

- Implement a trivial markup language distinguishing between headers and paragraphs in Lua
- Hand over the abstract syntax tree (AST) from Lua to rust
- Represent AST content in EPUB & PDF output for one header and one paragraph
  (line breaking and page breaking algorithms are missing)
- Publish the bibparser crate
- Propose a basic data model for a "heuristic Teχ parser" in rust
- I downloaded 50% of all arxiv papers to have them available as test suite. That costed me 105€.
- Introduce the typho.org website and update the github typho organization

Decisions

- A "requirement analysis" document is impractical. I would like to design my resources on digital typesetting like a university course ("introduction to digital typesetting"). And then design some slides and illustrations to communicate my approach using the terminology established in this course.

- “Text documents as trees” is a pragmatic abstraction I assume for upcoming work
- Unicode is a wonderful effort. Understand “character” according to the Unicode definition.
- I would love to see more FOSS standards in use. But PDF as supported standard is too strong and is going to be our main output format for many years to come.
- WebAssembly is an interesting platform.
- Good error messages and concurrency can give a huge advantage compared to existing solutions.

Milestone achieved: 2022-07-22

Milestone 0.2

Goals

- I need resources to communicate my approaches to the public.
- I need to (somewhat) finish my requirements notes and delve into implementation details for core algorithms.
- I need to properly understand how hooks can work to separate content from program logic in a user-friendly manner.
- I need to understand in which ways PDF allows encoding of images

Education

- Write down the details of "text documents as trees"
- Give a rudimentary talk (planned 2023-02-07) within a local community and thus design slides which provide a basis for "introduction to digital typesetting"
- Register a club in Austria with the generic goal to advance the state of digital typesetting
- Publish my requirement notes
- "Getting started" blog posts for the following typesetting software:
  - opTeχ
  - LuaLaTeχ
  - SILE
  - speedata publisher
  - typst

Implementation

- Typho design package encompassing logo, a banner artwork, and a designer guideline. Settle with this design for the next 5 years.
- Publish my strop program to experiment with Unicode operations
- Publish my litua program to transform content trees with Lua
- My prototype shall be extended with the representation of JPEG files in the PDF backend
- Finish and publish typherr for error message representation

Decisions

- I think to progress earlier towards a published software, because the market demands it

Future plan: Milestone 0.3

Goals

- Rudimentary OTF font support.
- Integrate HarfBuzz into my prototype in a meaningful way.
- We need progress in terms of core algorithms.
    - Working hyphenation algorithm.
    - Working line breaking algorithm.
    - Working page breaking algorithm.

Education

- Describe the HarfBuzz API in a blogpost
- “Getting started” blog posts for the following typesetting software:
    - Teχ
    - LaTeχ
    - ConTeχt
    - Patoline
    - LibreOffice
    - Scribus
    - MS Word
    - Adobe Illustrator

Implementation

- Hyphenation
    - Franklin Liang’s algorithm (for EN text) is available as a separate rust crate
    - Franklin Liang’s algorithm is integrated into the prototype
- Line breaking
    - Knuth-Plass algorithm (for EN text) is available as a separate rust crate
    - Knuth-Plass is integrated into the prototype
- A draft of my markup language paper is available (examples for all markup languages, list of all implementations, implementations installed on my computer, comparison of data models)

Rough future plan: Milestone 0.4

Goals

- A prototype exists, but its source code is quick & dirty. We want to have a publicly shared implementation.
- The implementation should illustrate the goals and approaches toward digital typesetting.
- To make this release pragmatic, the features are limited to the English language [without any notations like math or music] and standard fonts [which need not be provided]. In essence, one needs to be able to generate documents for text without any complex features.
- All components to convert a text file into these formats need to be publicly available.
- Publish my research on markup languages.

Education

Implementation

- An incomplete asciidoctor parser implementation (which handles all elements, but need not pass entire testsuite) is available

Rough future plan: Milestone 0.5

Goals

- Mathematical typesetting progress & support
    - Take over MathML’s data model to my prototype.
    - Write a parser which is able to recognize MathML and (most) Teχ formula syntax and store it in the data model.
    - Ability to output the data model in MathML or Teχ.
    - Testsuite: The goal is a heuristic parser for mathematical formulas to recognize 70% of formulas in arxiv.org papers.

Education

Implementation

Decisions

Rough future plan: Milestone 0.6

Goals

- Internationalized error messages in the core software
- Provide internationalization/localization utilities in the programming layer.

Education

Implementation

- strategy blog post for i18n of error messages in the CLI

Decisions

Future plan: Milestone 1.0

Goals

- We provide an implementation which allows us to build a financial basis for a sustainable future
- The published software satisfies the following requirements:
    - The software accepts one of the following text inputs:
        - CommonMark text documents
        - AsciiDoc text documents
        - XML documents with a XSD schema, I am providing
        - A text document object is provided through the API and satisfies the structure, I specify in the data model specification
    - Proper CLI error messages (four representations w.r.t. colors/Unicode) with a data model for various usecases occuring in typho
    - Proper metadata encoding (for example, title, author, license, and description)
    - Basic hyphenation, line breaking, and page breaking algorithms in place
    - Layout engine:
        - L/R alignment for text and images
        - indentation (for example, for blockquotes)
        - lists and tables
        - footnotes
        - page headers and footers
    - Simple table of contents, index, and bibliography generation support
    - The programming layer allows to apply the majority of Unicode operations specified by ICU
    - Font support:
        - OTF subsetting support
        - validation and feedback about availability / choice of Unicode scalar from the files
    - A rudimentary module system is designed to allow extensibility in the future
    - The software accepts the following font and media input files:
        - OTF font files

        - PNG, JPEG and SVG image files
    - The software is capable of generating the following output formats:
        - PDF
        - EPUB
        - HTML5 / webpage
        - manpage
        - plain text
    - The software can generate 500 pages per second for an average document (roughly 0.7 images/page, 1.6 section headers per page)

Education

Implementation

Decisions

Future plan: Milestone 2.0

Goals:

- Take the Wikipedia article about writing system X as an excerpt limited to one page written in X and typeset it. Then print this as a book.