fountain-parser README

version 0.1.0.0

Synopsis

`fountain-parser` is a small parser library for the FOUNTAIN screenplay format, fully supporting 1.1 version syntax and producing a simple, easy to grok AST.

`fountain-parser` is written in HASKELL and it uses the MEGAPARSEC library for parsing.

Disclaimer

Currently, this is *pre-alpha* software, not yet usable in productive form.

Motivation

The *Developers* section of the Fountain site provides a link to a parsing library in OBJECTIVE C. This presents a portability issue: while there *are* projects that make it possible to bridge OBJECTIVE C and HASKELL, they're platform- or framework-specific. That library informs this project in matching the different Fountain entities even as it uses different parsing methods.

Prospective Related Projects

`fountain-parser` aims to power a series of command-line utilities for conversion from FOUNTAIN to a series of convenient formats (like `.tex`) without intervention from thirds.

My software already supports Fountain

The *Apps* section of the FOUNTAIN site lists software that also imports or exports the format. There's a caveat: most are *cloud-based* and/or *proprietary*. By favoring (mostly) open formats, *fountain-parse* allows integration into many FLOSS tools, enabling entirely non-proprietary workflows and helping the creation of compound documents such as production bibles.

Implementation Specifics

In general, the library parser is rather lenient, allowing liberal spacing and recognizing UNICODE codepoints. Languages without uppercase/lowercase distinction must resort to *power-user characters* for case-dependent items such as transitions ('>') and character names ('@').

- As per the syntax guide:
  - This library expects FOUNTAIN text to be encoded in UTF-8.

  - Tabs are converted into **four** spaces.
  - Your line spacing is respected.
  - Initial spaces are ignored everywhere except in action lines.
  - A line with two spaces doesn't count as an empty line.
- All parsing functions expect `Text` inputs. File I/O is left to the application or framework.
- Varying-width UNICODE spaces are either converted into regular spaces or suppressed if they're hairline- or zero-width.
- Vertical tabs and form-feed characters are interpreted as line changes. For vertical spacing, use multiple blank lines and/or the FOUNTAIN form feed character sequence ("===") instead.
- The parser keeps everything: notes, boneyards, sections and synopses. Some possible conversion targets have analogues to those, so it might be desirable to preserve them.

Tentative Grammar

The following is an attempt to formalize the syntax in ABNF, drawing from the syntax guide and OBJECTIVE C implementation. Note that parsing actually occurs at the line level so the grammar should be considered a guide.

Building

GHC 9.6.7 and CABAL 3.0 (or greater) are required to compile the library and run the tests (*not implemented yet.*)

The project uses the `GHC2021` language default. While it might be possible to compile it in earlier versions than 9.6.7, this default is only available since 9.2.1, constituting a hard limit.

Some of the included scripts require `make`, `sed` and other similar utilities usually found in LINUX or LINUX-like environments (e.g., MSYS2. For WINDOWS users, it is recommended to use the GHCUP distribution, allow the installer script to deploy MSYS2 and then install the development packages.)

Contact

Please create an issue if you find a bug.

I can be reached at *10951848+CübQfJúdãhsLîòn* ă(t) *users/noreply/gīthụb/cȯm* (without diacritics and replacing slashes by periods.)