CPSC 498 Proposal: RBTEX

Steven Rosendahl

1 Abstract

Modern LATEX distributions include a tool called `lualatex` that allows users to dynamically produce content via use of Lua code. Unfortunately, the Lua standard libraries do not have as much functionality as other popular scripting languages, such as Ruby. The goal of this project is to incorporate Ruby into LATEX in a manner similar to `lualatex` , but with the power and simplicity of Ruby over Lua.

2 Motivation and Description

The current `lualatex` specification allows users to use several environments for writing and running Lua scripts. In addition, `lualatex` provides a built in library called `tex` that allows output to be printed straight to the LATEX document. The library, called RBTEX, will provide similar functionality through a gem called `rbtex`. In addition, the entire Ruby standard library will be available for use; RBTEX documents that need to interact directly with the system will most likely need to be compiled using the `--shell-escape` flag.

To use the library, users will need to have a Ruby version in the path. The code will be pre-processed, and inserted directly into the TEX code before `pdflatex` is called on the document. In addition, users will be provided with several ways in which to interact with Ruby from the TEX environment:

1. `inrbtex{}`: This command will provide a way for a use to execute one line of Ruby code at a time, or call a predefined function.
2. `rbtex{}`: This command will provide a way to write multiple lines of Ruby code inside the LATEX document. Any functions defined in this section will be globally defined, so they can be called in the `inrbtex{}` environment and in other `rbtex{}` environments. This code will be pulled verbatim from the LATEX file during the pre-processing stage.
3. `frbtex{}`: It may sometimes be convenient for a user to write an external Ruby file and call it from the LATEX file, rather than writing the code. The `frbtex{}` macro will allow an external script to be loaded into the document. The pre-processor will copy the provided Ruby file into the LATEX document, and will assume that all modules, classes, functions, and variables are globally defined.

The library will come with a program called `rbtex` that complies the provided LATEX document, much like the `luatex` command.

The program will work in four steps. It will first pre-process the TEX file, scanning for the appropriate environments. The ruby code will be ripped out and stored in an `.aux` file. The next step will order the code in the `.aux` file, and attempt to produce a `.rb` file from the provided LATEX document. In the third step, the code in the `.rb` file will be run using the `ruby` command specified in the user's `$PATH` variable. Finally, the post-processor will capture any output specified by the `tex` module in the `rbtex` gem, and place it into the LATEX document. From there, `pdflatex` will take over.

The pre-processor and post-processor will be written in C++ to allow for quick speed when parsing out the Ruby code. Shell script (UNIX) and Batch files (Win) will be provided for calling the program. Standard `pdflatex` flags will be available (they will be simply passed to the `pdflatex` command at the appropriate time). The final version of the program will be accessible through the shell command `rblatex tex_file.tex --shell-escape` and the Windows equivalent.

3 Tentative Schedule

January 19	Proposal and initial git setup
January 26	Research luatex implementation
February 2	Begin writing pre/post-processor
February 9	Continue writing pre/post-processor
February 16	Continue writing pre/post-processor
February 23	Test pre/post-processor
March 1	Test pre/post-processor
March 8	Build TEX bindings
March 15	Build TEX bindings
March 22	Test entire package
March 29	Build C program to include in TEX package
April 5	Clean and final test code
April 12	Extra space to be used as needed
April 19	Extra space to be used as needed