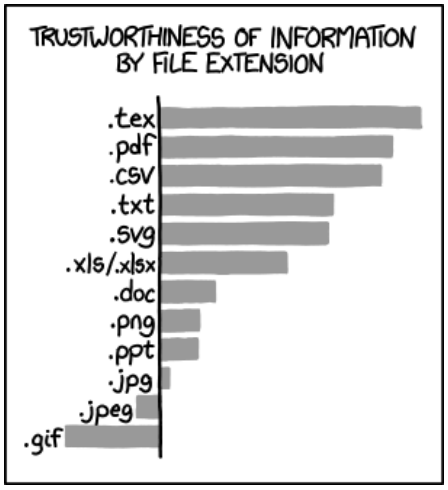
A Beginner's Haskell Workflow in Emacs

Rushi Shah

5 December 2015



When it came time to write college essays, I decided to write them all in LaTeX. Is LaTeX a bit overkill for something that normal people would use Microsoft Word for? Yeah probably. But

- I wanted to learn LaTeX anyways
- My editor configuration is so much better than the un-customizable Microsoft Word
- I can easily version-control my essays with git since `.tex` files aren't rich-text like `.docx` files

So I continued on my merry way and happily wrote my essays in LaTeX, created a git repository, used Sublime Text's LaTeXTools for the build system, etc. But the issue arose when I tried to send my essays to my family to look over. My sister wasn't a fan of the pdf format I had sent her, but she humored me. My parents, on the other hand, refused to look at anything other than Microsoft Word files. If I wanted to continue to use LaTex, I had to find a way to convert the files to word before sending them to my family.

1 Pandoc

The obvious answer was Pandoc. It is terrific and easily handled the task of converting `.tex` to `.docx`. And best of all, it is written in Haskell! But it is still a command line utility, so I

needed a way to set it up to automatically make the conversion when needed. I could have created a gulp task that watched the files for changes and made the conversion. But there was no point in converting every time I saved because I would work on the file for a while before I was ready to send it to my family. At most, I would need a converted word file every time I made a change significant enough to commit to git (commit early and often, kids!). So ideally, the converting would be bundled with the action of adding a file, commiting it, and pushing to my Github remote (which is unfortunately still a private repository).

2 The Bash Script

The most elegant solution I could think of was to just create a bash alias for that action.

The first step was a wrapper function that would take as many files as needed and convert them all to word format, saving the resulting file in a folder that can be added to the `.gitignore`. The feature of converting an arbitrary number of files at a time is unnecessary, but doesn't hurt.

```
latToDoc(){
    #Usage: `latToDoc FILE_NAME`
    mkdir -p word_format #In case the folder doesn't exist yet
    for arg in "\dollar@"
    do
        echo "Converting \dollararg"
        pandoc -f latex -t docx \dollararg.tex -o word_format/\dollararg.docx
        open word_format
        echo "Converted \dollararg"
    done
}
```

Then, I needed a quick function that would convert the file, add it, commit it with the provided commit message, and push it to the remote.

```
pushAndConvert(){
    #Usage: `pushAndConvert FILE_NAME "COMMIT_MESSAGE"`

    latToDoc \dollar1

    git add \dollar1.tex \dollar1.pdf
    echo "Added \dollar1.tex and \dollar1.pdf"

    git commit -m "\dollar2"
    echo "Committed with message: \dollar2"

    git push origin master
    echo "Pushed to master"
}
```

Now all I do is `pushAndConvert accept_me_please "Doubled bribery offer"`. Simple enough, right?