Multiple documents from one source

LaTeX for lecturers and teachers

Leo Arnold

latex@arney.de

09. October 2012

Typical approaches and roads to ruin

- typesetting the problems in `Sheet01.tex` and the solutions in `Sheet01-solutions.tex`
  - tedious to maintain congruence once you start changing the order of problems
  - changes in notation have to be updated in two separate files
  - students require problem to also be provided right above their solution
- typesetting the problems in `Sheet01.tex`, then copying this file to `Sheet01-solutions.tex` and filling in the solutions
  - Now students nag about incongruencies, and all the above other disadvantages remain
  - even worse: Now the problem is typeset twice and both versions have to be kept in sync too
- Typesetting problems and solutions in `Sheet01.tex`
  - needs some kind of switch to in/exclude solutions
  - pay close attention not to distribute the version with solutions to students first

Roadmap

1. Designing a unified workflow
   - Requirements
   - Implement a switch to in/exclude specific content
   - Produce all versions in one go
   - Harness the power of freeware math software

Roadmap

Requirements

- Only **one** source document
  - this is natural for typesetting the solution sheet
  - no duplicate texts, no forgotten copy & paste updates
  - maintain congruence of notation (Replace-all function)
- Easy reuse of parts in future courses
  - allow for divide & conquer using `\include`
  - automate the generation of the solution of some math problems

Roadmap

Clearifing the use cases

Our problem sheet needs to come in three different versions

student	just the problems
teacher	the problems and their solutions
corrector	problems, solutions and notes on grading

Solution: Use `comment.sty` by Victor Eijkhout

Credits: pointed out to me by Rolf Niepraschk

What comment.sty does

provides two simple commands

- `\includecomment{foobar}`
  Defines the environment `foobar` whose content will be included
- `\excludecomment{foobar}`
  Defines the environment `foobar` whose content will be ignored

```
\includecomment{wisdom}
\excludecomment{nonsense}

Confucius says:

\begin{wisdom}
Man who chases two rabbits
catches none.
\end{wisdom}

\begin{nonsense}
Donald Knuth invented WinWord.
\end{nonsense}
```

```
Confucius says:
Man who chases two rabbits
catches none.
```

Applying what we just learned

*	`\includecomment{problem}`
student	`\excludecomment{solution}`
	`\excludecomment{howtograde}`
teacher	`\includecomment{solution}`
	`\excludecomment{howtograde}`
corrector	`\includecomment{solution}`
	`\includecomment{howtograde}`

Let's start our own little uniflow.cls

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{uniflow}

\RequirePackage{comment}
  \includecomment{problem}

\DeclareOption{student}{%
  \excludecomment{solution}
  \excludecomment{howtograde}}
\DeclareOption{teacher}{%
  \includecomment{solution}
  \excludecomment{howtograde}}
\DeclareOption{corrector}{%
  \includecomment{solution}
  \includecomment{howtograde}}
\DeclareOption*{%
  \PassOptionsToClass{\CurrentOption}{article}}

\ExecuteOptions{corrector} % set default view
\ProcessOptions\relax

\LoadClass{article}
```

Roadmap

1. Designing a unified workflow
   - Requirements
   - Implement a switch to in/exclude specific content
   - Produce all versions in one go
   - Harness the power of freeware math software

Batch document production

One call of LaTeX will only produce a single document file, so what can we do?

- shell script
- MakeFile
- pseudo-executables
- Something with LUA in it.

but then again

- requires extra script file
- requires non-TeX programs
- may not be platform independent
- requires beyond-TeX knowledge

so ain't there a genuine pdfLaTeX approach?

Escaping to the Shell

pdfLaTeX can actually execute commands on the shell

```
\write18{echo "Hello World!"}
```

which also means pdfLaTeX can call pdfLaTeX … way to go!

Giving LaTeX access to the shell is a gateway for exploits. Hence `\write18` is disabled by default. Only the switch

```
pdflatex --shell-escape
```

will enable shell access.

Rethinking command line calls

Normal usage

```
pdflatex Sheet01.tex
```

is the same as

```
pdflatex "\input{Sheet01.tex}"
```

so we can insert a little code preceding the actual document

```
pdflatex "\gdef\conditionmacro{student} \input{Sheet01.tex}"
```

hence we need to adapt our `uniflow.cls` to react to `\conditionmacro` (later).

A neat trick by Ulrike Fischer (TeX.SX #5265)

Sheet01.tex

```
\ifx\conditionmacro\undefined
  \immediate\write18{%
    pdflatex --jobname=\jobname-student
    "\gdef\string\conditionmacro{student}
     \string\input\space\jobname"
  }
  \expandafter\stop
\fi
...
```

- `\immediate`: execute command immediately upon parse
- `\expandafter`: postpone expanding `\stop`, read `\fi` first
- `\stop`: Don't read any further

A neat trick by Ulrike Fischer (TeX.SX #5265)

Sheet01.tex

```
\ifx\conditionmacro\undefined
  \immediate\write18{%
    pdflatex --jobname=\jobname-student
    "\gdef\string\conditionmacro{student}
     \string\input\space\jobname"
  }
  \expandafter\stop
\fi
...
```

So what happens here?

- `pdflatex --shell-escape Sheet01.tex` reads until `\fi` and stops
- So far no output
- `\write18` has called `pdflatex` on the shell
- The `\ifx`-block is ignored because `\conditionmacro` is now defined
- This will produce `Sheet01-students.pdf`

Adapting the class file

```
[...]
\newif\ifuniFirstOrder\uniFirstOrdertrue

\DeclareOption{batch}{%
  [\immediate\write18-Trick]
}

\DeclareOption{student}{%
  \ifuniFirstOrder
    \excludecomment{solution}
    \excludecomment{howtograde}
  \fi
}
[...]
\ifx\conditionmacro\undefined
  \ExecuteOptions{corrector} % set default view
\else
  \ExecuteOptions{\conditionmacro}
  \uniFirstOrderfalse
\fi
\ProcessOptions\relax
```

Roadmap

1. Designing a unified workflow
   - Requirements
   - Implement a switch to in/exclude specific content
   - Produce all versions in one go
   - Harness the power of freeware math software

For simplicity of comprehension ...

- we will not pursue adapting `uniflow.sty`
- we will focus on implementing Ulrike's trick
- references to survey articles are provided
- if you can't understand the math – feel it
- this section is Linux-only – get used to it...

Algebra using Sage

About Sage and SageTEX

- `http://www.sagemath.org`
- Sage is developed primarily for Ubuntu
- The Windows version is a VirtualBox containing Ubuntu with Sage
- Hence SageTEX will not work using a Windows-LATEX
- Be sure to use `sagetex.sty` that came with your Sage installation

How to compile:

```
pdflatex SageExample.tex
sage SageExample.sagetex.sage
pdflatex SageExample.tex
```

Credits: Günter Rau
"SageTEX", DTK 2011-1, p. 17ff

Statistics using R

About R and Sweave

- R is the freeware alternative to commercial SPlus
- Highly used in psychological and educational research
- LaTeX-Plugin Sweave:
  `http://www.statistik.lmu.de/~leisch/Sweave/`
- Used NoWeb markup, hence the file extension .Rnw
- Hence bash completion is a pitfall when using Ulrike's trick

How to compile:

```
R CMD Sweave SweaveExample.Rnw
pdflatex SweaveExample.tex
```

Credits: Uwe Ziegenhagen
"Datenanalyse mit Sweave, LaTeXund R", DTK 2010-4, p. 35ff