TeX and controlled access to information

Boris Veytsman

Abstract

While we in the TeX community strive to make information open, there are cases when controlling access to information is legitimate. We do not want to publish our passwords, medical histories or other sensitive details. Sometimes the information is not confidential, but different audiences can have different needs: consider students' vs. teachers' versions of a textbook.

There are two aspects of this problem. Output-level control means that we have a single source which can produce different output files depending on compilation options. Source-level control means having different versions of "sources" obtained from the same master file.

In this paper we discuss tools for both of these approaches and their implementation in a TeX system.

1 Introduction

One popular technology activist slogan is "Information wants to be free", attributed to Stewart Brand [1]. It seems almost sacrilegious to use free (as in speech) tools like TeX and friends to hide information. However, this slogan as stated should refer to scientific, technical or cultural information only: obviously there is plenty of private information that we do not want to be "free". For instance, GNU cryptography tools of high quality do exist.

There are several cases when information hiding may be desired. First, the information can be uninteresting. Imagine you create a formal statement of work for a software development project. This statement includes financial information (how much each part of the project costs), technical information (which libraries and languages will be used), project milestones, etc. One can imagine several audiences with different needs. Some (e.g., executives) would want to read the document in its entirety. The financial team may not want to be bothered by technical details, while the development team may be bored by the financial details. Therefore we need several overlapping versions of the output intended for the different audiences. We will call this *output level control*.

A related, but different problem arises when we have several authors, and want to hide parts of the document from some of its authors. This possibility may seem rather exotic, but consider a report having classified and unclassified information. Among its authors could be experts that lack the clearance to see some parts of the document. Public and secret parts could be interspersed: for example, we might have classified footnotes to the unclassified body text. We want the authors to be able to work on their parts without endangering the overall secrecy level. We will call this *input level control*.

In this paper we discuss both of these issues and our solutions.

2 Existing LaTeX solutions for output level control

LaTeX provides a number of options to control the output of the document. The most direct one is the `\include` command and the `\includeonly` mechanism. We can separate text into parts intended for different audiences, and for each audience use only the relevant parts.

However, each included part starts a new page in the document, which does not allow interspersed and overlapping parts intended for different audiences. Also, putting different parts in different files might be confusing, especially when there are more than two different audiences.

Some of these problems can be eliminated with the `\input` command, which does not start a new page. There is no `\inputonly` command in LaTeX analogous to `\includeonly`, but it is easy to emulate it, for example,

```
\newif\iffinancial
...
\iffinancial\input{cost_table.tex}\fi
```

Still, the requirement to store content in different files is onerous.

The *comment* package [2] eliminates many of these problems. It allows the user to define comment-like environments and selectively output them with `\includecomment` and `\excludecomment` commands.

A similar approach is used by the *beamer* package [3], where one can include or exclude notes for presentations.[1]

When the package described in the next section was published on CTAN, Robin "ypid" Schneider informed me that another package with the same functionality, *tagging* by Brent Longborough, already exists [4]. I regret to say I simply missed this package in my search.

3 Output level control: a new package *multiaudience*

The new package *multiaudience* [5] tries to provide

[1] I am grateful to Joseph Wright and other participants of TUG'15 for this remark.

a clean interface for output level control. Its main concept is *audience*. A document can define several audiences using the \SetNewAudience macro, for example,

```
\SetNewAudience{admins}
\SetNewAudience{devs}
\SetNewAudience{execs}
```

This code defines three audiences, *admins*, *devs*, and *execs*.

The document can have one and only one *current audience*, which is stored in the \CurrentAudience macro. The author may define it using the command \DefCurentAudience

```
\DefCurrentAudience{admins}
```

or with just a TEX \def:

```
\def\CurrentAudience{admins}
```

The latter possibility allows the user to define the current audience outside the document (from the command line), for example

```
pdflatex -jobname devs \
 '\def\CurrentAudience{devs}\input{master.tex}'
```

This approach has the advantage of keeping the tex file (master.tex in this case) clean, and generating different versions on the fly.

The heart of the package is \showto macro. It has two arguments: the comma-separated list of audiences and the text to show to these audiences, for example,

```
\showto{admins}{This text
  is visible to admins only.}
```

The command can be nested:

```
\showto{admins, devs}{This text is visible
  to admins and devs. \showto{devs}{This
  text is visible to devs only.} This text
  is visible to admins and devs again.}
```

In the example above the text is visible to admins and devs with the exception of the italicized text, which is visible to devs only.

A variant of the \showto macro uses exclusion rather than inclusion logic: if the first argument starts with a minus sign, it defines the audiences which will *not* see the information:

```
\showto{-, execs}{This text is  visible to
  everybody but execs.}
```

The package provides the environment shownto with the similar syntax and semantics:

```
  \begin{shownto}{admins,devs}
    This text is visible to admins and devs
    \begin{shownto}{-, admins}
      This text is visible to devs only
    \end{shownto}
    This text is visible to admins and
    devs again.
  \end{shownto}
```

There are also special commands like \Footnote, making selectively visible footnotes:

```
  We have a special footnote
  command.\Footnote{admins}{This
  footnote is for admins only.}
```

and section-like environments:

```
\begin{Subsection}{admins}[Short title]{Long
    title}
This subsection is visible only to admins
\end{Subsection}
```

Moreover, the user can define new commands and section-like environments with a simple interface:

```
\DefMultiaudienceCommand{\Footnote}%
   {\footnote}
\NewMultiaudienceSectionEnv{Subsection}%
   {\subsection}
```

The current implementation is very simple: basically we evaluate a TEX boolean \if@MULTAU@shown using the first argument of the \showto command, and typeset or not the second argument with the construction

```
\if@MULTAU@shown#2\fi
```

The actual implementation is slightly more involved since we need to check whether the first argument is "-", and accordingly to select inclusion or exclusion logic. See the source code [5] for the full details.

Petr Olšák wrote a plain TEX implementation even before my talk at TUG'15 was finished [6].

Of course this simplicity has its drawbacks: you cannot use verbatim construction in the second argument of \showto command. There are a number of workarounds here, see, e.g. [7].

4 Source level control

Source level control should solve two problems, one being easy, and another being slightly more difficult:

1. Extracting the material intended for different authors.
2. Accommodating the changes made by any author.

The second problem involves the following situation. Suppose a non-cleared author makes a change to the unclassified part of a document. We need to be able to accommodate her changes in the master document, which includes classified material.

Perl script *srcredact* [8] solves both these problems. Its interface is a simplified interface of the *docstrip* program [9]: we have special comment lines in

```
\documentclass{article}
\begin{document}
\title{A Letter to the Secretary
  of the Treasury}
\author{Mark Twain}
\date{Riverdale-on-the-Hudson, October 15, 1902}
\maketitle

%</ALL>
%<*uppercase|nobonds>
THE HON. THE SECRETARY OF THE TREASURY,
WASHINGTON, D.~C.:
%<*ALL>
%</uppercase|nobonds>
\textsc{the hon. the secretary of the treasury,
  washington, d.~c.:}
%<*ALL>

Sir,---Prices for the customary kinds of winter
fuel having reached an altitude which puts them
out of the reach of literary persons in
straitened  circumstances, I desire to place
with you the following order:

%</nobonds>
Forty-five tons best old dry government bonds,
suitable for furnace, gold 7 per cents.,
1864, preferred.
%<*ALL>

Twelve tons early greenbacks, range size,
suitable for cooking.

Eight barrels seasoned 25 and 50 cent postal
currency, vintage of 1866, eligible for
kindlings.

Please deliver with all convenient despatch
at my house in Riverdale at lowest rates for
spot  cash, and send bill to

Your obliged servant,

Mark Twain, Who will be very grateful,
and will vote right.
\end{document}
```

Figure 1: Example of a TEX file for *srcredact* tool

the TEX file (*guards*): either `%<*name1|name2|...>` or `%</name1|name2|...>`. The first one switches on the inclusion of text, while the second one switches it off. The special name `ALL` stands for all names. By default the text is considered to be enclosed in an `<*ALL>`/`</ALL>` pair.

An example input file is shown in Figure 1. It defines three versions of the same document, `default`, `uppercase` and `nobonds`. We can extract these three versions. If one version, say, `nobonds`, is changed, we can incorporate the changes into the main file and re-generate the three versions, as shown in Figure 2.

5 Conclusions

Information separation, hiding and control provide an interesting problem for document creation tools. TEX, being a programmable tool based on a text interface, is quite useful for solving it.

Acknowledgements

This work was partially supported by Neadwerx, Inc. and the US Consumer Financial Protection Bureau.

References

[1] R. Polk Wagner. Information wants to be free: Intellectual property and the mythologies of control. *Columbia Law Rev.*, 103:995–1034, 2003.

[2] Victor Eijkhout. *The comment package*, October 1999. `http://www.ctan.org/pkg/comment`.

[3] Till Tantau, Joseph Wright, and Vedran Miletić. *The beamer class*, March 2015. `http://www.ctan.org/pkg/beamer`.

[4] Brent Longborough. *tagging.sty. A package for document configuration*, August 2011. `http://www.ctan.org/pkg/tagging`.

[5] Boris Veytsman. *Generating multiple versions of a document for different audiences from the same source*, August 2015. `http://www.ctan.org/pkg/multiaudience`.

[6] Petr Olšák. Hidden text from unprivileged readers. `http://petr.olsak.net/opmac-tricks-e.html#showif`, July 2015.

[7] Timothy Van Zandt, Denis Girou, Sebastian Rahtz, and Herbert Voß. *The fancyvrb package. Fancy Verbatims in LATEX*, May 2010. `http://www.ctan.org/pkg/fancyvrb`.

[8] Boris Veytsman. *A tool for redacting the sources*, August 2015. `http://www.ctan.org/pkg/srcredact`.

[9] Frank Mittelbach, Denys Duchier, Johannes Braams, Marcin Woliński, and Mark Wooding. *The DocStrip program*, November 2014. `http://www.ctan.org/pkg/docstrip`.

◇ Boris Veytsman
Systems Biology School and Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030 USA
`borisv (at) lk dot net`
`http://borisv.lk.net`

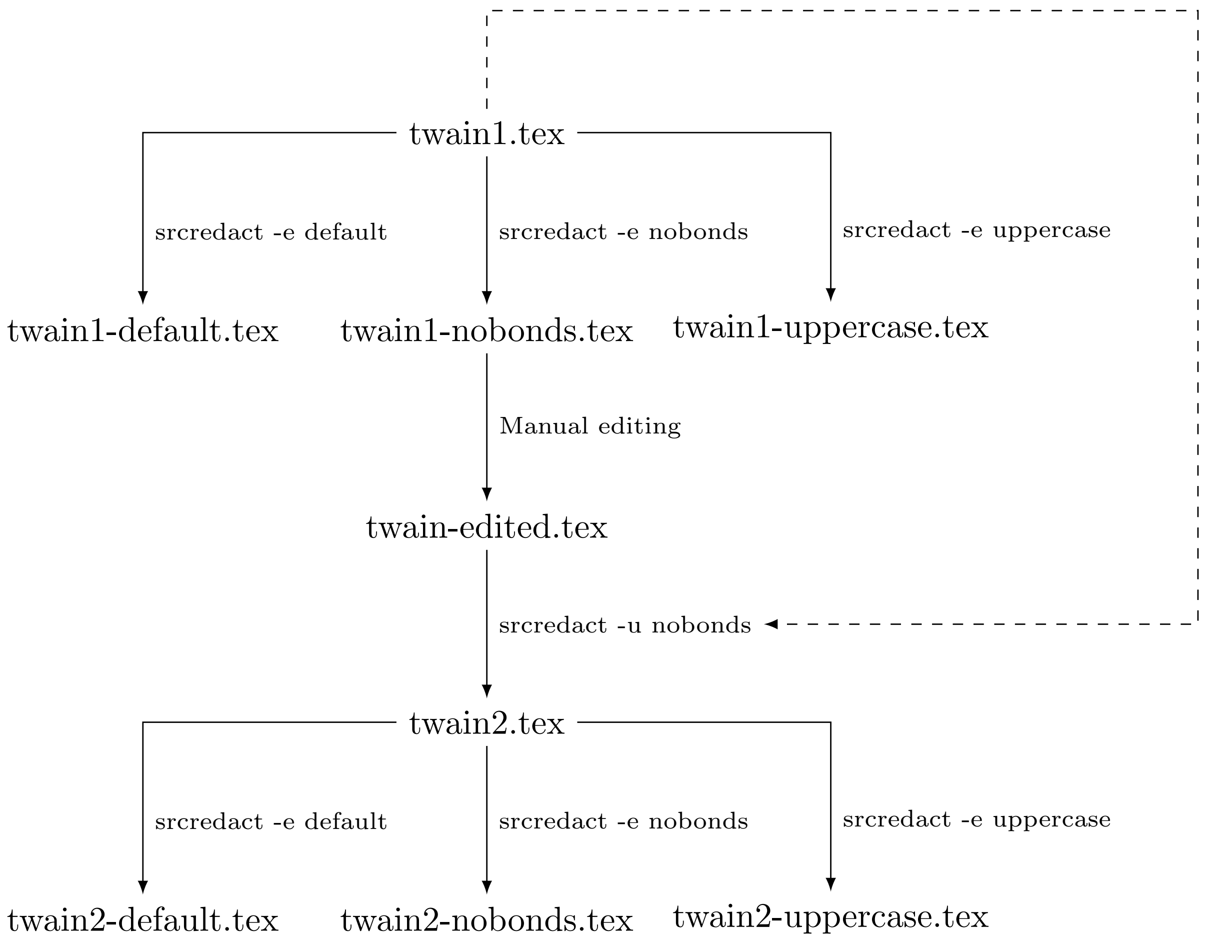

Figure 2: Document workflow with optional redactions