Recording Errata in LaTeX Documents[*]

Michael Kohlhase
Computer Science, Jacobs University, Bremen, Germany
`http://kwarc.info/kohlhase`

April 14, 2015

Abstract

This package provides a simple infrastructure for recording errata in LaTeX documents. This allows to maintain an updated version of the document (with all errors corrected) and automatically generate an errata document highlighting the difference to the published version.

1 Introduction

The life-cycle of a document does not end with its publication. After that, errors will be discovered, and have to be managed. The best way to do this is by marking errata in the text and generating the errata document from that. In printed books, errata documents came in the form of an errata sheet, which listed errors and was inserted into the book manually, with the expectation that they would be corrected in a later printing. In electronic documents where there is no printing cycle, errata documents (or sections) are only sensible if there is a "release cycle", e.g. for standards.

The development version of this package can be found on GitHub at `https://github.com/KWARC/LaTeX-errata`

Acknowledgements Thanks to Johan Schroll-Fleischer for comments and bug reports.

2 The User Interface

`show` `hide` `mark` `margins` `foots` The `errata` package can be called with a variety of package options. The `show` and `hide` options govern the general visibility of the errata markup. If `hide` is specified (the default, if no option is given), then errata markup is totally invisible in the formatted document. The `mark`, `margins`, and `foots` options give finer control together with their negative variants `nomark`, `nomargins`, and `nofoots`. If the `mark` (`margins`, `foots`) option is given, then erratum markup (margin decorations, explanatory footnotes) are generated. The `show` option (the default) is equivalent to `mark,margins,foots`, the `hide` option to `nomark,nomargins,nofoots`.

`record` If the `record` option is specified, then errata are written to a special file ⟨*jobname*⟩`-errata.tex` so that they can be included in an errata document (see Section 2.3).

There are two kinds of errata, short ones or extended ones. We will go into them in the next two subsections. All the errata macros take optional KeyVal arguments that specify the metadata for the errata. `date=` `reported-by=` We can specify the date when the erratum was found, with the `date` key and the person that reported the erratum with with the `reported-by` key. Consider for instance the metadata for the erratum in Example 2; The date format follows the ISO 8601 norm: it is of the form ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩, where ⟨*YYYY*⟩ is the 4-digit year, ⟨*MM*⟩ the 2-digit month, and

[*]Version v0.4 (last revised 2015/04/13)

⟨*DD*⟩ the two-digit day of the month. The format of the `reported-by` field is a free-form text[1]. We can specify an identifier for the erratum using the `id` key, so that we can refer back to it later. `id=` Finally, we can use the `type` key to specify the type (e.g. "clarification") of an erratum. `type=`

2.1 Short Errata

Short errata can be recorded by the marking up the editing operations needed for the corrections. The following example in Example 1 shows all editing operations, which we will detail now:

```
Here we have \erratumAdd{forgotten word}{three} errata
in one \erratumDelete{superfluous phrase}{darned} long
\erratumReplace{translated}{Zeile}{line}
```

Example 1: Some short errata

`\erratumAdd` marks up a correction by adding some text. The first argument holds the explanation of the intended correction and the second one the new text. `\erratumAdd`

`\erratumDelete` does the same for a correction by deleting some text; here the second argument holds the deleted text. While the `\erratumAdd` will copy the contents of the second argument to the result document, `\erratumDelete` does not. `\erratumDelete`

`\erratumReplace{[`⟨*keys*⟩`]`⟨*desc*⟩`}{`⟨*old*⟩`}{`⟨*new*⟩`}` replaces ⟨*old*⟩ with ⟨*new*⟩ in the result document. `\erratumReplace`

All of these macros mark the location of the errata in the margin and document the changes in footnote-like structures. For instance, the text fragment above would be rendered as

Here we have $[\text{three}]^1_a$ errata in one $[]^2_d$ long $[\text{line}]^3_r$ Err(1) Err(2) Err(3)

2.2 Extended Errata

Extended errata group multiple editing operations into a coherent group via the `erratum` environment. The first argument of this environment is an explanation as for the short errata above. The `erratum` environment provides local versions of the editing markup macros, which behave like those, but lack the first (explanation) argument, which is already given in the environment that contains them. `erratum`

```
\begin{erratum}[date=2006-07-19,reported-by=Michael Kohlhase]{old should be new}
this is a test of a long erratum
\begin{enumerate}
  \item We can replace \eReplace{oldtext}{newtext}
  \item and \eAdd{new text}
  \item and finally delete old text\eDelete{alltogether}
  \end{enumerate}
\end{erratum}
```

Example 2: An extended erratum with local correction markers

This text fragment would be rendered as BErr(4)

this is a test of a long erratum

[1]Remember to wrap the value in curly braces if the name contains a comma
[1]ERRATUM! forgotten word (added text)
[2]ERRATUM! superfluous phrase (deleted "darned")
[3]ERRATUM! translated (original text was: "Zeile")
[4]ERRATUM: OLD SHOULD BE NEW

1. We can replace [newtext]$_r^{4:1}$
2. and [new text]$_a$
3. and finally delete old text[]$_d^{4:2}$

EErr(4)

The erratum environment should also be used in situations where the error occurs in an environment, where normal TEX/LATEX processing is suspended, e.g. a verbatim environment. In this case, we can use it to attach correction information via the environment, but do not use the local change documentations.[1] EdN:1

2.3 Generating Statistics and Errata Documents

\erratamessage Putting the macro \erratamessage just before the \end{document} will generate a message with cardinality information for the errata into the log file.

\erratumItem errata Errata can be marked up using the \erratumItem macro in the theerrata environment. The \erratumItem takes two arguments, the first is a reference to where the erratum occurred, and the second one the explanation.

\printerrata The \printerrata command allows to print the errata for another document. This command is useful when generating errata documents for published works. Say we have a book with a driver file thebook.tex, into which we have incorporated errata markup using the infrastructure detailed above. Then we have a new document called e.g. theerrata.tex which has the form given in Figure 3. Note that we have used \printerrata{thebook} to include the errata notices generated from thebook.tex.

```
\documentclass{article}
\usepackage[hide]{errata}
\title{Errata for The Book}
\begin{document}
\maketitle
\begin{abstract} This document tracks the errata in The Book. \end{abstract}
\section{Introduction}
 ... The errata of The Book are tracked in this document, whose newest version can be
 found at \url{.../berrata.pdf}. A version of The Book that contains all errata
corrections (and markup of what changed) can be found at \url{.../book.pdf}.

In the following we will tabulate the errata in document order. Their location will be
referenced by the section they appear in rather than the page number, since we do not
expect the former to change in the errata correction process.

\section{The Errata in The Book}
\printerrata{thebook}
\end{document}
```

Example 3: A Sample Errata Document

In the errata document in Figure 3 we postulate that we keep an updated version of The Book online[2] using the infrastructure provide by the errata package. In the updated version of thebook.tex, it can be useful to tabulate the errata as well, e.g. in a section in the appendix. This can be done by the \PrintErrata command. Note that this command needs to close the errata file thebook-errata.tex therefore we need a \newpage to clear the queue of waiting \writes before thebook-errata.tex can be loaded (otherwise we may be missing the errata from the last page). \printerrata

[4:1] was: oldtext
[4:2] deleted: alltogether
[1] EdNote: what do we do in floats? document
[2] And indeed it is good practice to do so if the copyright agreement with the publisher allows this.

3 Limitations

The support offered by this package is still relatively basic. There are many possible extensions I would like, but do not have the time for right now. It would be nice to have

1. the erratum marks configurable by a user-defined macro (e.g. for using colors instead of square brackets)
2. erratum markup show the old version next to the new ones in place (MS Word and google docs do a nice job of this in change mode)
3. hyperlinking errata marks to the printed errata

If you want to contribute, please contact the author or make a pull request at `https://github.com/KWARC/LaTeX-errata`

4 The Implementation

The implementation is rather standard. We first set up the options for the package. The main switches are `\ifmargins`, `\iffoots`, and `\ifmark`, which govern the visibility of the errata markup and the annotations in the margins and the footnotes.

```
⟨*package⟩
\newif\ifmargins\marginstrue
\newif\iffoots\footstrue
\newif\ifmark\marktrue
\newif\ifrecord\recordfalse
```

the next step is to declare the package options, they just set the switches accordingly.

```
\DeclareOption{show}{\marginstrue\footstrue}
\DeclareOption{hide}{\marginsfalse\footsfalse}
\DeclareOption{margins}{\marginstrue}
\DeclareOption{foots}{\footstrue}
\DeclareOption{nomargins}{\marginsfalse}
\DeclareOption{nofoots}{\footsfalse}
\DeclareOption{mark}{\marktrue}
\DeclareOption{nomark}{\markfalse}
\DeclareOption{record}{\recordtrue}
\ProcessOptions
```

This ends the package setup code, so we can come to the implementation of the functionality of the package.

We first make sure that the `Keyval` package is loaded.

```
⟨*package⟩
\RequirePackage{keyval}[1997/11/10]
```

and then define the actions that are undertaken when the keys are encountered. Here this is very simple, we just define an internal macro with the value, so that we can use it later.

```
\define@key{erratum}{id}{\def\erratum@id{#1}}
\define@key{erratum}{type}{\def\erratum@type{#1}}
\define@key{erratum}{date}{\def\erratum@date{#1}}
\define@key{erratum}{reported-by}{\def\erratum@reported-by{#1}}
```

4.1 Recording Errata

The next step is to set up two counters for the errata.

```
\newcounter{erratum}
\newcounter{erratum@note}[erratum]
```

If the `record` option is specified, we open a file for writing out the errata.

```
\ifrecord\newwrite\errata@file
\immediate\openout\errata@file=\jobname-errata.tex
\AtEndDocument{\closeout\errata@file}\fi
```

`\record@erratum` The `\record@erratum` macro just writes its argument to the errata file together with referencing information

```
\def\ErratumRef{\@ifundefined{thechapter}{}{\arabic{chapter}.}%
\@ifundefined{thesection}{}{\ifnum\value{section}>0{}\arabic{section}\fi}%
\@ifundefined{thesubsection}{}{\ifnum\value{subsection}>0.\arabic{subsection}\fi}%
\@ifundefined{thesubsubsection}{}{\ifnum\value{subsubsection}>0.\arabic{subsubsection}\fi}}
\def\record@erratum#1{\ifrecord\protected@write\errata@file{}%
{\string\erratumItem{\ErratumRef}{#1}}\fi}
```

`\erratumItem` EdN:2 For the errata themselves we use the following macro, which treats them as items in a `description` environment[2]

[2] EdNote: would be better to number errata and treat them like glossary entries: 1. explanation 3.4

```
\def\erratumItem#1#2{\item[#1] #2}
```

\printerrata \PrintErrata The `\printerrata` inputs the errata file for the path given in its argument. Its variant `\PrintErrata` macro closes the errata file if necessary and calls `\printerrata` for its own errata.

```
\def\printerrata#1{\IfFileExists{#1-errata.tex}{\begin{errata}\input{#1-errata}\end{errata}}{}}
\def\PrintErrata{\ifrecord\immediate\closeout\errata@file\fi\printerrata\jobname}
```

errata The `errata` environment wraps the errata items. Currently, this is just a `description` environment.

```
\newenvironment{errata}{\begin{description}}{\end{description}}
```

4.2 Short Errata

[3] EdN:3

\erratumAdd

```
\newcommand{\erratumAdd}[3][]% keyvals, explanation, new
{\setkeys{erratum}{#1}\stepcounter{erratum}\record@erratum{#2}%
\ifmargins\marginpar{Err(\arabic{erratum})}\fi\immediate\typeout{Erratum!}%
\ifmark[\fi #3\ifmark]$_a^{\arabic{erratum}}$\fi%
\iffoots\footnotetext[\value{erratum}]{{\scshape{Erratum!}}%
\@ifundefined{erratum@type}{}{(\erratum@type)} #2 (added text)}\fi}
```

\erratumDelete

```
\newcommand{\erratumDelete}[3][]% keyvals, explanation, old
{\setkeys{erratum}{#1}\stepcounter{erratum}\record@erratum{#2}%
\ifmargins\marginpar{Err(\arabic{erratum})}\fi\immediate\typeout{Erratum!}%
\ifmark[]$_d^{\arabic{erratum}}$\else\ignorespaces\fi%
\iffoots\footnotetext[\value{erratum}]{{\scshape{Erratum!}}%
\@ifundefined{erratum@type}{}{(\erratum@type)} #2 (deleted ``#3'')}\fi}
```

\erratumReplace

```
\newcommand{\erratumReplace}[4][]% keyvals, explanation, old, new
{\setkeys{erratum}{#1}\stepcounter{erratum}\record@erratum{#2}%
\ifmargins\marginpar{Err(\arabic{erratum})}\fi\immediate\typeout{Erratum!}%
\ifmark[\fi #4\ifmark]$_r^{\arabic{erratum}}$\fi%
\iffoots\footnotetext[\value{erratum}]{{\scshape{Erratum!}}%
\@ifundefined{erratum@type}{}{(\erratum@type)} #2 (original text was: ``#3'')}\fi}
```

4.3 Extended Errata

erratum The `erratum` environment is for extended errata. It steps the counters, marks the margins (if desired) and sets up local macros

```
\newenvironment{erratum}[2][]% keys, explanation
{\setkeys{erratum}{#1}\stepcounter{erratum}%
\edef\new@number{\theerratum}\message{Erratum \new@number!}%
\iffoots\footnotetext[\value{erratum}]{{\scshape{Erratum}%
\@ifundefined{erratum@type}{}{(\erratum@type)}: #2}}%
\ifmargins\marginpar{BErr(\new@number)}\fi\fi%
\record@erratum{#2}%
```

\eAdd This macro just passes through the argument, and delimits it, so that the extent of the insertion can be seen.

```
\def\eAdd##1{\ifmark[\fi ##1\ifmark]$_a$\fi}%
```

[3] EdNote: these could do with a bit of code refactoring

`\eDelete` This macro just passes through the argument, and explains what was deleted in a footnote.

```
\def\eDelete##1{\erratum@mark\ifmark[]$_d^{\@thefnmark}$\else\ignorespaces\fi%
\iffoots\@footnotetext{deleted: ##1}\fi}%
```

`\eReplace` This macro just passes through the second argument, delimits it and marks the replacement in a footnote.

```
\def\eReplace##1##2{\erratum@mark\ifmark[\fi ##2\ifmark]$_r^{\@thefnmark}$\fi%
\iffoots\@footnotetext{was: ##1}\fi}%
\ignorespaces}
```

The end part of the `erratum` environment is almost trivial, it only marks the margin with the end mark.

```
{\ifmargins\marginpar{EErr(\new@number)}\fi}
```

`\erratum@mark` This macro is used by the local macros of the `erratum` environment, it sets the footnote label by redefining the LaTeX-internal `\@thefnmark` macro appropriately.

```
\def\erratum@mark{\stepcounter{erratum@note}{}%
\def\@thefnmark{\arabic{erratum}:\arabic{erratum@note}}}
```

4.4 Generating Statistics

`\ednotemessage` The `\ednotemessage` makes use of the counter `ednote` and generates a message.

```
\def\ednotemessage{\ifnum\value{erratum}>0\typeout{}%
\typeout{This document contains \arabic{erratum} Errata; see \jobname-errata.tex!}%
\typeout{}\fi}
⟨/package⟩
```

5 The Errata of this Document