Recording Errata in LaTeX Documents*

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

November 4, 2007

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.[1]

EdNote(1)

2 The User Interface

`hide` 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 `margins` and `foots` options give finer control. If the `margins` (`foots`) option is not given, then margin decorations (explanatory footnotes) are not generated. the `hide` option is equivalent to `margins` and `foots`.

`margins/foots`

`hide`

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

`date=`

`reported-by=`

`id=`

`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:

`\erratumAdd` `\erratumAdd` marks up a correction by adding some text. The first argument holds the expla-

*Version Errata (last revised 2006/11/121v0.3)

[1] EdNote: describe some more

[1] Remember to encasulate the value in curly braces if the name contains a comma

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

Example 1: Some short errata

nation of the intended correctionand the second one the new text.

`\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 [three]$^{1}_{a}$ errata in one []$^{2}_{d}$ long [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. We can replace [newtext]$^{4:1}_{r}$
2. and [new text]$_{a}$
3. and finally delete old text []$^{4:2}_{d}$

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.[2] EdNote(2)

[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
[4:1] was: oldtext
[4:2] deleted: alltogether
[2] EDNOTE: what do we do in floats? document

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 the`errata` 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

\printerrata 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).

[2] And indeed it is good practice to do so if the copyright agreement with the publisher allows this.

3 The Implementation

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

```
⟨*package⟩
\newif\ifmargins\marginsfalse
\newif\iffoots\footsfalse
\newif\ifrecord\recordfalse
```

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

```
\DeclareOption{show}{\marginstrue\footstrue}
\DeclareOption{hide}{}
\DeclareOption{margins}{\marginstrue}
\DeclareOption{foots}{\footstrue}
\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}}
```

3.1 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` For the errata themselves we use the following macro, which treats them as items in a `description` environment[3] EdNote(3)

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

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

\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}}
```

3.2 Short Errata

EdNote(4) [4]

\erratumAdd

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

\erratumDelete

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

\erratumReplace

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

3.3 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}}\marginpar{BErr(\new@number)}\fi
\record@erratum{#2}
```

\eAdd

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

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

\eDelete

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

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

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

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

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

The end part of the `erratum` environement 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}}}
```

3.4 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⟩

1 `old` should be `new`

2 forgotten word

2.1 superfluous phrase

2.1 translated

Change History

v0.1
General: First Version with Documentation 1
v0.2
General: More Documentation and Keywords Management 1
v0.3
General: adding type Keyword 1

4 The Errata of this Document

1 `old` should be `new`

2 forgotten word

2.1 superfluous phrase

2.1 translated