The *changes*-package

Manual change markup — version 4.2.1

July 15, 2021

Ekkart Kleinod

@   ekleinod@edgesoft.de

1 Introduction

This package provides means for manual change markup.

Any comments, thoughts or improvements are welcome. The package is maintained at *gitlab*, please see

```
https://edgesoft.de/projects/changes/
```

for links to source code access, bug and feature tracker, etc. If you want to contact me directly, please send an email to ekleinod@edgesoft.de. Please start your email subject with `[changes]`.

> The changes-package allows the user to manually markup changes of text, such as additions, deletions, or replacements. Changed text is shown in a different color; deleted text is striked out. Additionally, text can be highlighted and/or commented. The package allows free definition of additional authors and their associated color. It also allows you to change the markup of changes, authors, highlights or comments.

Here is a short example of change markup:

[EK 1] missing word

[EK 2] For the fun of it.

> This is new text. In this sentence, I replace a goodbad^{EK} word. And, to sum up the text changes, there is another obsolete^{EK} word to delete. Furthermore, text can be highlighted ^{EK} or just commented.

Parallel to this manual is a folder "examples" which contains an extensive collection of example files, both LaTeX and PDF files. Please refer to these examples for inspiration and first problem solving.

2 Using the *changes*-package

In this section a typical use case of the *changes*-package is described. You can find the detailed description of the package options and new commands in Section 4.

We start with the text you want to change. You want to markup the changes for each author individually. Such a change markup is well-known in WYSIWYG text processors such as *LibreOffice*, *OpenOffice*, or *Word*.

The *changes*-package was developed in order to support such change markup. The package provides commands for defining authors, and for marking text as added, deleted, or replaced. Additionally, text can be highlighted or commented. In order to use the package, you should follow these steps:

1. use *changes*-package
2. define authors
3. markup text changes
4. highlight and comment text
5. typeset the document with LaTeX
6. output list of changes
7. remove markup

Use *changes*-package

In order to activate change management, use the *changes*-package as follows:

```
\usepackage{changes}
```

respectively

```
\usepackage[<options>]{changes}
```

You can use the options for defining the layout of the change markup. You can change the layout after using the *changes*-package as well.

For detailed information please refer to Section 4.1 and Section 4.6.

Define authors

The *changes*-package provides a default anonymous author. If you want to track your changes depending on the author, you have to define the needed authors as follows:

```
\definechangesauthor[name=<name>, color=<color>]{<id>}
```

Every author is uniquely identified through his or her id. You can give every author an optional name and/or color.

For detailed information please refer to Section 4.5.

Markup text changes

Now everything is set to markup the changed text. Please use the following commands according to your change:

for added text:

`\added[id=<id>, comment=<comment>]{<new text>}`

for deleted text:

`\deleted[id=<id>, comment=<comment>]{<old text>}`

for replaced text:

`\replaced[id=<id>, comment=<comment>]{<new text>}{<old text>}`

Stating the author's id and/or a comment is optional.

For detailed information please refer to Section 4.2.

Highlight and comment text

Maybe you want to highlight orcomment some text?

highlight text:

`\highlight[id=<id>, comment=<comment>]{<text>}`

comment text:

`\comment[id=<id>]{<comment>}`

Stating the author's id and/or a comment for highlights is optional.

For detailed information please refer to Section 4.3.

Typeset the document with LaTeX

After marking your changes in the text you are able to display them in the generated document by processing it as usual with LaTeX. By processing your document the changed text is layouted as you stated by the corresponding options and/or special commands.

Output list of changes

You can print a list of changes using:

```
\listofchanges[style=<style>, title=<title>, show=<type>]
```

The list is meant to be the analogon to the list of tables, or the list of figures.

Stating the style is optional, default is `style=list`. In order to print a quick overview of the number and kind of changes of every author, use the option `style=summary` or `style=compactsummary`. Show only specific changes by using the `show` option.

By running LaTeX the data of the list is written into an auxiliary file. This data is used in the next LaTeX run for typesetting the list of changes. Therefore, two LaTeX runs are needed after every change in order to typeset an up-to-date list of changes.

For detailed information please refer to Section 4.4.

Remove markup

Often you want to remove the change markup after acknowledging or rejecting the changes. You can suppress the output of changes with:

```
\usepackage[final]{changes}
```

In order to remove the markup from the LaTeX files, you have to remove the commands by hand or use the script by Yvon Cui. You find the script `pyMergeChanges.py` in the directory:

```
<texpath>/scripts/changes/
```

The script removes all markups either keeping or rejecting the change. You can select or deselect markup from removal using the interactive mode by starting the script without options.

For detailed information please refer to Section 5.

3 Limitations and possible enhancements

The *changes*-package was carefully programmed and tested. Yet the possibility of errors in the package exists, you might encounter problem during use, or you might miss functionality.

You can find a list of th emost important known problems and possible solutions in Section 6. Please refer to the section first if your problem is known and is a solution exists. More errors, problems, and solutions are provided at:

```
https://edgesoft.de/projects/changes/
```

or

```
https://gitlab.com/ekleinod/changes/-/issues
```

You can write me an email too, please send it to ekleinod@edgesoft.de. In that case, please start your email subject with `[changes]`.

Change markup of texts works well, it is possible to markup whole paragraphs. You cannot markup:

– figures
– tables
– headings
– some commands
– multiple paragraphs (sometimes)

You can try putting such text in an extra file and include in with `input`. This works sometimes, give it a try. Kudos to Charly Arenz for this tip.

If you experience errors about already defined macros, please see option `commandnameprefix`, Section 4.1.3.

4 User interface of the *changes*-package

This section describes the user interface of the *changes*-package, i.e. all options and commands of the package. Every option and new command is described. If you want to see the options and commands in action, please refer to the examples in

`<texpath>/doc/latex/changes/examples/`

The example files are named with the used option respectively command.

4.1 Package Options

`\usepackage[<options>]{changes}`

The package options control the behavior of the overall package, i.e. all markup commands.

The following options are defined:

4.1.1 draft

`\usepackage[draft]{changes}` ~ `\usepackage{changes}`

The `draft`-option enables markup of changes. The list of changes is available via `\listofchanges`. This option is the default option, if no other option is selected.

The *changes* package reuses the declaration of `draft` in `\documentclass`. The local declaration of `final` overrules the declaration of `draft` in `\documentclass`.

4.1.2 final

`\usepackage[final]{changes}`

The `final`-option disables markup of changes, only the correct text will be shown. The list of changes is disabled, too.

The *changes* package reuses the declaration of `final` in `\documentclass`. The local declaration of `draft` overrules the declaration of `final` in `\documentclass`.

4.1.3 commandnameprefix

`\usepackage[commandnameprefix=<strategy>]{changes}`

The `commandnameprefix` option sets the prefixing strategy for the markup commands. This is useful if another package already defined commands, e.g. `\comment` or `\highlight`.

Per default an error is raised if a command is already defined and no prefixing takes place (option not given or set to `none`).

If a prefix strategy is set, the command in question is prefixed with "ch". The strategy determines which commands are prefixed.

This option only provides prefixed names for the markup commands:

– `\added` → `\chadded`
– `\deleted` → `\chdeleted`
– `\replaced` → `\chreplaced`
– `\highlight` → `\chhighlight`
– `\comment` → `\chcomment`

The following strategies for *commandnameprefix* are provided:

none   no prefix, a command already defined raises an error (default strategy)

ifneeded	if a command is already defined, *changes* prefixes this command and raises a warning. Depending on the commands already defined, the document will contain a mix of prefixed and not prefixed markup commands. This is mostly used if only `\comment` or `\highlight` are already defined and you mainly want to use the change commands `\added`, `\deleted`, and `\replaced`.
always	all commands are prefixed, an according message is written to the log

Examples

```
\usepackage[commandnameprefix=none]{changes} ~ \usepackage{changes}
\usepackage[commandnameprefix=ifneeded]{changes}
\usepackage[commandnameprefix=always]{changes}
```

4.1.4 markup

```
\usepackage[markup=<markup>]{changes}
```

The markup option chooses a predefined visual markup of changed text. The default markup is chosen if no explicit markup is given. The markup chosen with markup can be overwritten with the more special markup options addedmarkup, deletedmarkup, commentmarkup, or highlightmarkup.

The following values for *markup* are defined:

default	default markup for added and deleted text, comments and highlighted text (default markup)
underlined	underlined for added text, wavy underlined for highlighted text, default for deleted text, and comments
bfit	bold added text, italic deleted text, default for comments and highlighted text
nocolor	no colored markup, underlined for added text, wavy underlined for highlighted text, default for deleted text and comments

Examples

```
\usepackage[markup=default]{changes} ~ \usepackage{changes}
\usepackage[markup=underlined]{changes}
\usepackage[markup=bfit]{changes}
\usepackage[markup=nocolor]{changes}
```

When changing from color markup to markup without color and vice versa, some errors occur if an auxiliary file exists. Please ignore the errors, they vanish in the second run.

4.1.5 addedmarkup

```
\usepackage[addedmarkup=<addedmarkup>]{changes}
```

The `addedmarkup` option chooses a predefined visual markup of added text. The default markup is chosen if no explicit markup is given. The option `addedmarkup` overwrites the markup chosen with `markup`.

The following values for *addedmarkup* are defined:

colored	no text markup, just coloring – example (default)
uline	underlined text – example
uuline	double underlined text – example
uwave	wavy underlined text – example
dashuline	dashed underlined text – example
dotuline	dotted underlined text – example
bf	bold text – **example**
it	italic text – *example*
sl	slanted text – *example*
em	emphasized text – *example*

The output of replaced text is a combination of added and deleted text, thus any change in their layout influences the layout of replaced text.

Examples

```
\usepackage[addedmarkup=colored]{changes} ~ \usepackage{changes}
\usepackage[addedmarkup=uline]{changes}
\usepackage[addedmarkup=bf]{changes}
```

4.1.6 deletedmarkup

```
\usepackage[deletedmarkup=<deletedmarkup>]{changes}
```

The `deletedmarkup` option chooses a predefined visual markup of deleted texty. The default markup is chosen if no explicit markup is given. The option `deletedmarkup` overwrites the markup chosen with `markup`.

The following values for *deletedmarkup* are defined:

sout	striked out text – ~~example~~ (default)
xout	crossed out text – example
colored	no text markup, just coloring – example
uline	underlined text – example
uuline	double underlined text – example
uwave	wavy underlined text – example

dashuline	dashed underlined text – <u>example</u>
dotuline	dotted underlined text – <u>example</u>
bf	bold text – **example**
it	italic text – *example*
sl	slanted text – *example*
em	emphasized text – *example*

The output of replaced text is a combination of added and deleted text, thus any change in their layout influences the layout of replaced text.

Examples

```
\usepackage[deletedmarkup=sout]{changes} ~ \usepackage{changes}
\usepackage[deletedmarkup=xout]{changes}
\usepackage[deletedmarkup=uwave]{changes}
```

4.1.7 highlightmarkup

```
\usepackage[highlightmarkup=<highlightmarkup>]{changes}
```

The `highlightmarkup` option chooses a predefined visual markup for highlighted text. The default markup is chosen if no explicit markup is given. The option `highlightmarkup` overwrites the markup chosen with `markup`.

The following values for *highlightmarkup* are defined:

background	markup by background color – example (default)
uuline	double underlined text – <u>example</u>
uwave	wavy underlined text – <u>example</u>

Examples

```
\usepackage[highlightmarkup=background]{changes} ~ \usepackage{
    changes}
\usepackage[highlightmarkup=uuline]{changes}
```

4.1.8 commentmarkup

```
\usepackage[commentmarkup=<commentmarkup>]{changes}
```

The `commentmarkup` option chooses a predefined visual markup for comments. The default markup is chosen if no explicit markup is given. The option `commentmarkup` overwrites the markup chosen with `markup`.

The following values for *commentmarkup* are defined:

<div style="float:left">**example comment**

example
comment</div>

`todo`	comment as todo note, which is not added to list of todos (default)
`margin`	comment in margin
`footnote`	comment as footnote[1]
`uwave`	wavy underlined text – example comment

Examples

```
\usepackage[commentmarkup=todo]{changes} ~ \usepackage{changes}
\usepackage[commentmarkup=footnote]{changes}
\usepackage[commentmarkup=uwave]{changes}
```

4.1.9 authormarkup

```
\usepackage[authormarkup=<authormarkup>]{changes}
```

The `authormarkup` option chooses a predefined visual markup of the author's identification. The default markup is chosen if no explicit markup is given.

The following values for *authormarkup* are defined:

`superscript`	superscripted text – text$^{\text{author}}$ (default)
`subscript`	subscripted text – text$_{\text{author}}$
`brackets`	text in brackets – text(author)
`footnote`	text in footnote – text[2]
`none`	no author identification

Examples

```
\usepackage[authormarkup=superscript]{changes} ~ \usepackage{
      changes}
\usepackage[authormarkup=brackets]{changes}
\usepackage[authormarkup=none]{changes}
```

---

[1]   example comment
[2]   author

4.1.10 authormarkupposition

```
\usepackage[authormarkupposition=<authormarkupposition>]{changes}
```

The `authormarkupposition` option chooses the position of the author's identification. The default value is chosen if no explicit markup is given.

The following values for *authormarkupposition* are defined:

right   right of the text – text^{author} (default)
left   left of the text – ^{author}text

Examples

```
\usepackage[authormarkupposition=right]{changes} ~ \usepackage{
    changes}
\usepackage[authormarkupposition=left]{changes}
```

4.1.11 authormarkuptext

```
\usepackage[authormarkuptext=<authormarkuptext>]{changes}
```

The `authormarkuptext` option chooses the text that is used for the author's identification. The default value is chosen if no explicit markup is given.

The following values for *authormarkuptext* are defined:

id   author's id – text^{id} (default)
name   author's name – text^{authorname}

Examples

```
\usepackage[authormarkuptext=id]{changes} ~ \usepackage{changes}
\usepackage[authormarkuptext=name]{changes}
```

4.1.12 defaultcolor

```
\usepackage[defaultcolor=<color>]{changes}
```

The `defaultcolor` option defines the default color for authors, including the color for the default (anonymous) author. You can use colors of the *xcolor* package.

The default color is *blue*.

Examples

```
\usepackage[defaultcolor=blue]{changes} ~ \usepackage{changes}
\usepackage[defaultcolor=magenta]{changes}
```

4.1.13 todonotes

```
\usepackage[todonotes=<options>]{changes}
```

Options for the *todonotes* package can be specified as parameters of the `todonotes`-option. Several options or options with special characters have to be put in curly brackets.

Examples

```
\usepackage[todonotes={textsize=tiny}]{changes}
```

4.1.14 truncate

```
\usepackage[truncate=<options>]{changes}
```

Options for the *truncate* package can be specified as parameters of the `truncate`-option. Several options or options with special characters have to be put in curly brackets.

Examples

```
\usepackage[truncate=hyphenate]{changes}
```

4.1.15 ulem

```
\usepackage[ulem=<options>]{changes}
```

Options for the *ulem* package can be specified as parameters of the `ulem`-option. Several options or options with special characters have to be put in curly brackets.

Examples

```
\usepackage[ulem=UWforbf]{changes}
\usepackage[ulem={normalem,normalbf}]{changes}
```

4.1.16 xcolor

```
\usepackage[xcolor=<options>]{changes}
```

Options for the *xcolor* package can be specified as parameters of the xcolor-option. Several options or options with special characters have to be put in curly brackets.

Examples

```
\usepackage[xcolor=dvipdf]{changes}
\usepackage[xcolor={dvipdf,gray}]{changes}
```

4.2 Change management

4.2.1 \added

```
\added[id=<id>, comment=<comment>]{<new text>}
```

The command \added marks newly added text. The new text is given in curly braces.

The optional argument contains key-value-pairs for author-id and comment. The author-id has to be defined using \definechangesauthor. If the comment contains special characters or spaces, use curly brackets to enclose the comment.

If a comment is given, the direct author markup at the changes text is omitted, because the author is printed in the comment.

Examples

```
This is \added{new} text.
This is \added[id=EK]{new} text too.
This is more \added[id=EK, comment={has to be in it}]{new} text.
This is the last \added[comment=anonymous]{new} text.
```

Result

[EK 3] has to be in it

[1] anony-mous

This is new text. This is new^{EK} text too. This is more new text. This is the last new text.

4.2.2 \deleted

```
\deleted[id=<id>, comment=<comment>]{<old text>}
```

The command `\deleted` marks deleted text. The deleted text is given in curly braces.

For the optional arguments see `\added` (Section 4.2.1).

Examples

```
This is \deleted{old} text.
This is \deleted[id=EK]{old} text too.
This is more \deleted[id=EK, comment={too old}]{old} text.
This is the last \deleted[comment=away]{old} text.
```

Result

[EK 4] too old

[2] away

This is ~~old~~ text. This is ~~old~~^{EK} text too. This is more ~~old~~ text. This is the last ~~old~~ text.

4.2.3 \replaced

```
\replaced[id=<id>, comment=<comment>]{<new text>}{<old text>}
```

The command `\replaced` marks replaced text. The new and the replaced text are given in this order in curly braces.

For the optional arguments see `\added` (Section 4.2.1).

The output of replaced text is a combination of added and deleted text, thus any change in their layout influences the layout of replaced text.

Examples

```
This is \replaced{new}{replaced} text.
This is \replaced[id=EK]{new}{replaced} text too.
This is more \replaced[id=EK, comment={better}]{new}{replaced} text
    .
This is the last \replaced[comment=improved]{new}{replaced} text.
```

Result

[EK 5] better

[3] improved

This is new~~replaced~~ text. This is new~~replaced~~^{EK} text too. This is more new ~~replaced~~ text. This is the last new~~replaced~~ text.

4.3 Highlighting and Comments

4.3.1 \highlight

\highlight[id=<id>, comment=<comment>]{<text>}

The command \highlight highlights text. The highlighted text is given in curly braces.

For the optional arguments see \added (Section 4.2.1).

Examples

```
This is \highlight{highlighted} text.
This is \highlight[id=EK]{highlighted} text too.
This is more \highlight[id=EK, comment={Good one.}]{highlighted}
    text.
This is the last \highlight[comment=remember]{highlighted} text.
```

Result

This is   highlighted   text.   This is   highlighted [EK] text too.   This is more   highlighted   text. This is the last   highlighted   text.

[EK 6] Good one.

[4] remember

4.3.2 \comment

\comment[id=<id>]{<comment>}

The command \comments adds a comment to the document. The comment is given in curly braces.

The command has only one optional argument: a key-value-pair for the author-id. The author-id has to be defined using \definechangesauthor.

The comments are numbered automatically, the number is printed in the comment.

Examples

```
This is \comment{Sure}commented text.
This is \comment[id=EK]{Correct.}commented text too.
```

Result

[5] Sure

[EK 7] Correct.

This is commented text. This is commented text too.

4.4 Overview of changes

4.4.1 `\listofchanges`

`\listofchanges[style=<style>, title=<title>, show=<type>]`

The command `\listofchanges` outputs a list or summary of changes. The first LaTeX-run creates an auxiliary file, the second run uses the data of this file. Therefore you need two LaTeX-runs for an up-to-date list of changes.

There are three optional arguments:

`style`	list style
`title`	individual title
`show`	markup types

style   The `style` argument defines the layout of the list of changes. Three styles are defined:

`list`	prints the list of changes like a list of figures (default)
`summary`	prints the number of changes grouped by author
`compactsummary`	same as `summary` but entries with count 0 are omitted

title   The `title` argument is used to change the title for the list. If you want to use special characters or spaces in the title, enclose it in curly braces.

show   The `show` argument defines which types of change markup are shown in the list of changes. You can combine the values using the | character. For example if you want to show all additions and deletions, use `show=added|deleted`.

The following values are defined:

`all`	show all types (default)
`added`	show only additions
`deleted`	show only deletions
`replaced`	show only replacements
`highlight`	show only highlights
`comment`	show only comments

Examples

```
\listofchanges
\listofchanges[style=list] ~ \listofchanges
\listofchanges[style=summary, title={My Summary}]
\listofchanges[title={List of comments}, show=comment]]
\listofchanges[style=compactsummary, show=added|deleted|replaced,
    title={Text changes}]]
```

4.5 Author management

4.5.1 \definechangesauthor

```
\definechangesauthor[name=<name>, color=<color>]{<id>}
```

The command \definechangesauthor defines a new author for changes. You have to define a unique author's id, special characters or spaces are not allowed within the author's id.

You may define a corresponding color and the author's name. If you do not define a color, blue is used.

The author's name is used in the list of changes and in the markup if you set the corresponding option.

The package predefines one anonymous author without id.

Examples

```
\definechangesauthor{EK}
\definechangesauthor[color=orange]{EK}
\definechangesauthor[name={Ekkart Kleinod}]{EK}
\definechangesauthor[name={Ekkart Kleinod}, color=orange]{EK}
```

4.6 Adaptation of the output

4.6.1 Values for markup command definitions

If you want to adapt the markup output, you can use any LaTeX-commands and special values resp. macros of the *changes* package. Some values or macros are specific for each command, they are described in the corresponding sections.

The following values and macros can be used in each command:

– any LaTeX-commands
– author's color can be used with color "authorcolor"
– boolean test if colored change output is needed "\IfIsColored"

I do not provide full access to all elements of the markup for every command in order to keep the macros simple. For example, the author's id is only available for \setcomment-markup.

The output of replaced text is a combination of added and deleted text.

4.6.2 \setaddedmarkup

 \setaddedmarkup{<definition>}

The command \setaddedmarkup defines the layout of added text. The default markup is colored text, or the markup set with the option markup respectively addedmarkup.

Values for definition:

– added text can be used with "#1"

The output of replaced text is a combination of added and deleted text, thus any change in their layout influences the layout of replaced text.

Examples

```
\setaddedmarkup{\emph{#1}}
\setaddedmarkup{+++: #1}
```

4.6.3 \setdeletedmarkup

```
\setdeletedmarkup{<definition>}
```

The command \setdeletedmarkup defines the layout of deleted text. The default markup is striked-out, or the markup set with the option markup respectively deletedmarkup.

Values for definition:

– deleted text can be used with "#1"

The output of replaced text is a combination of added and deleted text, thus any change in their layout influences the layout of replaced text.

Examples

```
\setdeletedmarkup{\emph{#1}}
\setdeletedmarkup{---: #1}
```

4.6.4 \sethighlightmarkup

```
\sethighlightmarkup{<definition>}
```

The command \sethighlightmarkup defines the layout of highlighted text. The default markup is via a background color, or the markup set with the option markup respectively highlightmarkup.

Values for definition:

– highlighted text can be used with "#1"

Examples

```
\sethighlightmarkup{\emph{#1}}
\sethighlightmarkup{{\IfIsColored{\color{authorcolor}}{}||: #1}}
```

4.6.5 \setcommentmarkup

\setcommentmarkup{<definition>}

The command \setcommentmarkup defines the layout of comments. The default markup is a margin note, or the markup set with the option markup respectively commentmarkup.

Values for definition:

– comment can be used with "#1"
– author's id can be used with "#2"
– author output (id or name) can be used with "#3"
– comment count of the autor can be used with counter "authorcommentcount"
– boolean test if author is anonymous "\IfIsAnonymous"

Examples

```
\setcommentmarkup{-- #1 --}
\setcommentmarkup{{\IfIsColored{\color{authorcolor}}{}#1}}
\setcommentmarkup{\IfIsAnonymous{#2}{}{\textbf{#3: }}#1}
\setcommentmarkup{[\arabic{authorcommentcount}] #1}
```

4.6.6 \setauthormarkup

\setauthormarkup{<definition>}

The command \setauthormarkup defines the layout of the author's markup in the text. The default markup is a superscripted author's text.

Values for definition:

– author output (id or name) can be used with "#1"

Examples

```
\setauthormarkup{(#1)}
\setauthormarkup{(#1)~---~}
\setauthormarkup{\marginpar{#1}}
```

4.6.7 \setauthormarkupposition

```
\setauthormarkupposition{<authormarkupposition>}
```

The command \setauthormarkupposition defines the position of the author's markup relative to the changed text. The default position is right of the changed text.

The following values for *authormarkupposition* are defined:

right	right of the text – text$^{\text{author}}$ (default)
left	left of the text – $^{\text{author}}$text

Examples

```
\setauthormarkupposition{right}
\setauthormarkupposition{left}
```

4.6.8 \setauthormarkuptext

```
\setauthormarkuptext{<authormarkuptext>}
```

The command \setauthormarkuptext defines the text for the author's markup. The default markup is the author's id.

The following values for *authormarkuptext* are defined:

id	author's id – text$^{\text{id}}$ (default)
name	author's name – text$^{\text{authorname}}$

Examples

```
\setauthormarkuptext{id}
\setauthormarkuptext{name}
```

4.6.9 \setanonymousname

```
\setanonymousname{<name>}
```

The command \setanonymousname sets the anonymous author's name. The default name is the language dependent equivalent of "anonymous".

This option is helpful if you are the only author and you want your name to be displayed at the changes.

Examples

```
\setanonymousname{Anonymous author}
\setanonymousname{My name}
```

4.6.10 \settruncatewidth

```
\settruncatewidth{<width>}
```

The command \settruncatewidth sets the width of the truncation in the list of changes to the given width. The default width is 0.6\textwidth.

Examples

```
\settruncatewidth{5cm}
\settruncatewidth{.3\textwidth}
```

4.6.11 \setsummarywidth

```
\setsummarywidth{<width>}
```

The command \setsummarywidth sets the width of the list of changes in summary style to the given width. The default width is 0.3\textwidth.

Examples

```
\setsummarywidth{3cm}
\setsummarywidth{.5\textwidth}
```

4.6.12 \setsummarytowidth

```
\setsummarytowidth{<text>}
```

The command \setsummarytowidth sets the width of the list of changes in summary style to the width of the given text.

Examples

```
\setsummarytowidth{Highlighted \qquad}
\setsummarytowidth{The longest text imaginable for the summary.}
```

4.6.13 \setlocextension

```
\setlocextension{<extension>}
```

The command \setlocextension sets the extension of the auxiliary file for the list of changes (loc-file[3]). The default extension is "loc".

In the example, the loc-file for "foo.tex" would be named "foo.listofchanges" resp. "foo.lochg" instead of the default name "foo.loc".

Examples

```
\setlocextension{listofchanges}
\setlocextension{lochg}
```

> Do not use a LaTeX standard file extension, such as "toc" or "lof", as this would collide with the normal LaTeX run.

4.6.14 \setsocextension

```
\setsocextension{<extension>}
```

The command \setsocextension sets the extension of the auxiliary file for the summary of changes (soc-file[4]). The default extension is "soc".

In the example, the soc-file for "foo.tex" would be named "foo.changes" resp. "foo.chg" instead of the default name "foo.soc".

Examples

```
\setsocextension{changes}
\setsocextension{chg}
```

> Do not use a LaTeX standard file extension, such as "toc" or "lof", as this would collide with the normal LaTeX run.

---

[3] "loc" stands for "list of changes".
[4] "soc" stands for "summary of changes".

4.7 Used packages

The *changes*-package uses already existing packages for it's functions. You will find detailed description of the packages in their distributions.

The following packages are always required and have to be installed for the *changes*-package:

etoolbox	provides an enhanced `\if`-commands, *bools*, or list operations
truncate	truncation of texts (used for list of changes)
xkeyval	provides key-value-lists for parameters
xstring	improves string operations

The following packages are sometimes required and have to be installed if used by the corresponding option:

todonotes	loaded if comments are layouted as todo notes (default markup)
ulem	loaded if text has to be striked, wavylined or exed out (default markup)
xcolor	loaded if colored text is used for markup (default markup)

5 Remove markup from file

In order to remove the markup from the LaTeX files, you have to remove the commands by hand or use the script by Yvon Cui. You find the script in the directory:

```
<texpath>/scripts/changes/
```

The script removes all markups either keeping or rejecting the change. You can select or deselect markup from removal using the interactive mode by starting the script without options.

The script requires *python3*.

Use the script as follows:

```
python pyMergeChanges.py [-arh] <Input File> <Output File>
Options:
        -a: accept all added, deleted and replaced
        -r: reject all added, deleted and replaced
        -h: remove all highlights
If no option is given, runs interactively.
```

Run the script with no options and files for a short help text:

```
python pyMergeChanges.py
```

Known issues:

– removes only markup that is used in one line, not markup that spans multiple lines

6 Known problems and solutions

This section contains known problems and their solutions as far as I know some. If your problem is not listed here, please see the issue tracker on gitlab if it contains your problem (a search exists):

`https://gitlab.com/ekleinod/changes/issues`

If your problem is not listed, please open a new issue for your problem. Describe your problem as specific as possible, if possible, include a small example file with the problematic behavior.

6.1 Special content

Change markup of texts works well, it is possible to markup whole paragraphs. You cannot markup:

– figures
– tables
– headings
– some commands
– several paragraphs (sometimes)

You can try putting such text in an extra file and include in with `input`. This works sometimes, give it a try. Kudos to Charly Arenz for this tip.

6.2 Footnotes and margin notes

There is a problem of typesetting footnotes or margin notes in special environments, such as tables or the *tabbing* environment. Avoid this type of markup when using these environments.

6.3 The *ulem* package

I am using the *ulem* package for striking out text as default. This causes problems with some commands and environments, e.g.

– in math mode
– when using the *siunitx* package
– when using the `\citet` or `\citep` command

In that case there are few good options, the best is to define the markup for deletions yourself and avoid the ulem package. See

– Section 4.1.6
– Section 4.6.3

6.4 Command already defined

Some packages use the same names for their commands as the *changes* package, in particular \comment and \highlight are not originally named commands.

In this case, *changes* may prefix its commands to avoid naming collisions. This is controlled by the commandnameprefix option, see Section 4.1.3 for the documentation.

In order for this to work, the *changes* package has to be loaded after the other packages or commandnameprefix=always must be selected.

7 Authors

Several authors contributed to the *changes*-package. Many bugs and problems were solved or their solution inspired via de.comp.text.tex. Thanks.

The authors are (in alphabetical order):

– Chiaradonna, Silvano
– Cui, Yvon
– Fischer, Ulrike
– Giovannini, Daniele
– Kleinod, Ekkart
– Le Garrec, Vincent
– Mittelbach, Frank
– Richardson, Alexander
– Voss, Herbert
– Wölfel, Philipp
– Wolter, Steve

8 Versions

For a list of versions and the changes within these version, please refer to

`https://gitlab.com/ekleinod/changes/blob/master/changelog.md`

Here you too find the implemented but not released changes for the new version.

If you are interested in planned new features, please see

`https://gitlab.com/ekleinod/changes/milestones`

9 Distribution, Copyright, License

This work consists of the files

```
source/latex/changes/changes.drv
source/latex/changes/changes.dtx
source/latex/changes/changes.ins
source/latex/changes/examples.dtx
source/latex/changes/regression.dtx
source/latex/changes/README
source/latex/changes/userdoc/*.tex
scripts/changes/pyMergeChanges.py
```

and the derived files

```
doc/latex/changes/changes.english.pdf
doc/latex/changes/changes.english.withcode.pdf
doc/latex/changes/changes.ngerman.pdf
doc/latex/changes/examples/changes.example.*.tex
doc/latex/changes/examples/changes.example.*.pdf
doc/latex/changes/regression/changes.regression.*.tex
doc/latex/changes/regression/changes.regression.*.pdf
tex/latex/changes/changes.sty
```

10 The documented sourcecode

The sourcecode is documented in English only. This is intended, please do not provide translations for the text below, just corrections or improvements.

```
1 ⟨∗changes⟩
```

10.1 Package information and options

Set needed LaTeX-format to LaTeX 2$_\varepsilon$, provide name, date, version. Type some information to the console.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{changes}[2021/07/15 v4.2.1 changes package]
4 \typeout{*** changes package 2021/07/15 v4.2.1 ***}
```

Package *xkeyval* provides options with key-value-pairs.

```
5 \RequirePackage{xkeyval}
```

Package *etoolbox* provides improved *if*, *bools* as well as a *list* operations. todo

```
6 \RequirePackage{etoolbox}
```

Package *xstring* provides improved string test and handling methods.

```
7 \RequirePackage{xstring}
```

\IfIsInList   Helper macro: tests if one of the pipe separated values is in the second list of pipe separated values (boolean or).

Corresponds to the if macros of *etoolbox*, so it can be used in it's boolean tests. Mainly the last two parameters are the results of *true* and *false* computation.

Declare list parsers.

```
8 \DeclareListParser{\dopsvlist}{|}
9 \DeclareListParser*{\forpsvlist}{|}
```

A temporary list and flag.

```
10 \newcommand{\Changes@tmplist}{}
11 \newbool{Changes@inlist}
```

Fill list with values, then check if one of the values of the first list is in the second one.

```
12 \newrobustcmd{\IfIsInList}[4]{%
13 \renewcommand{\Changes@tmplist}{}%
```

I could not get \forpsvlist{\listadd\Changes@tmplist}{#2} to work if #2 is a macro, so I used the code below.

Thanks to Ulrike Fischer for the fix of the macro call of \dopsvlist with \expandafter.

To make sure \do can be renewed, we define it first, some macros, such as \chapter undefine it.

```
14 \def\do{}%
15 \renewcommand*{\do}[1]{%
16 \listadd{\Changes@tmplist}{##1}%
17 }%
18 \expandafter\dopsvlist\expandafter{#2}%
```

End fo the workaround for \forpsvlist.

```
19 \setbool{Changes@inlist}{false}%
20 \renewcommand*{\do}[1]{%
21 \ifinlist{##1}{\Changes@tmplist}%
22 {\setbool{Changes@inlist}{true}}%
23 {}%
24 }%
25 \expandafter\dopsvlist\expandafter{#1}%
26 \ifbool{Changes@inlist}%
27 {#3}%
28 {#4}%
29 }
```

10.1.1 Package options

Option draft, *default* is *true*.

```
30 \newbool{Changes@optiondraft}
31 \setbool{Changes@optiondraft}{true}
32 \DeclareOptionX{draft}{
33 \setbool{Changes@optiondraft}{true}
34 \typeout{changes-option '\CurrentOption'}
35 }
```

Option final, sets draft to *false*.

```
36 \DeclareOptionX{final}{
37 \setbool{Changes@optiondraft}{false}
38 \typeout{changes-option '\CurrentOption'}
39 }
```

Declare storage for authormarkup option and store option value or set to default value *superscript*.

```
40 \newcommand{\Changes@optioncommandnameprefix}{none}
41 \DeclareOptionX{commandnameprefix}{
42 \ifblank{#1}
43 {}
44 {
45 \IfIsInList{#1}{none|always|ifneeded}
46 {\renewcommand{\Changes@optioncommandnameprefix}{#1}}
47 {\PackageWarning{changes}{commandnameprefix '#1' unknown, using '\Changes@optioncommandnamep
48 }
49 \typeout{changes-option 'commandnameprefix=\Changes@optioncommandnameprefix'}
50 }
```

Declare storage for markup options, they are set by the markup option but can be changed with the more special options, therefore they have to be declared at this place. Replacement markup is a combination of added and deleted markup, thus there is no special markup storage.

```
51 \newcommand{\Changes@optionaddedmarkup}{colored}
52 \newcommand{\Changes@optiondeletedmarkup}{sout}
53 \newcommand{\Changes@optionhighlightmarkup}{background}
54 \newcommand{\Changes@optioncommentmarkup}{todo}
```

Option `markup`, sets markup options accordingly.

```
55 \newcommand{\Changes@optionmarkup}{default}
56 \DeclareOptionX{markup}{
57 \ifblank{#1}
58 {}
59 {
60 \IfIsInList{#1}{bfit|default|nocolor|underlined}
61 {\renewcommand{\Changes@optionmarkup}{#1}}
62 {\PackageWarning{changes}{markup '#1' unknown, using '\Changes@optionmarkup'}}
63 }
64 \IfStrEq{\Changes@optionmarkup}{default}
65 {
66 % nothing to do, included for symmetry and potential later use
67 }
68 {}
69 \IfStrEq{\Changes@optionmarkup}{underlined}
70 {
71 \renewcommand{\Changes@optionaddedmarkup}{uline}
72 \renewcommand{\Changes@optionhighlightmarkup}{uwave}
73 }
74 {}
75 \IfStrEq{\Changes@optionmarkup}{bfit}
76 {
```

```
77 \renewcommand{\Changes@optionaddedmarkup}{bf}
78 \renewcommand{\Changes@optiondeletedmarkup}{it}
79 }
80 {}
81 \IfStrEq{\Changes@optionmarkup}{nocolor}
82 {
83 \renewcommand{\Changes@optionaddedmarkup}{uline}
84 \renewcommand{\Changes@optionhighlightmarkup}{uwave}
85 }
86 {}
87 \typeout{changes-option 'markup=\Changes@optionmarkup'}
88 }
```

Option addedmarkup, stored or set to default value *colored*.

```
89 \DeclareOptionX{addedmarkup}{
90 \ifblank{#1}
91 {}
92 {
93 \IfIsInList{#1}{bf|colored|dashuline|dotuline|em|it|sl|uline|uuline|uwave}
94 {\renewcommand{\Changes@optionaddedmarkup}{#1}}
95 {\PackageWarning{changes}{addedmarkup '#1' unknown, using '\Changes@optionaddedmarkup'}}
96 }
97 \typeout{changes-option 'addedmarkup=\Changes@optionaddedmarkup'}
98 }
```

Option deletedmarkup, stored or set to default value *sout*.

```
 99 \DeclareOptionX{deletedmarkup}{
100 \ifblank{#1}
101 {}
102 {
103 \IfIsInList{#1}{bf|colored|dashuline|dotuline|em|it|sl|sout|uline|uuline|uwave|xout}
104 {\renewcommand{\Changes@optiondeletedmarkup}{#1}}
105 {\PackageWarning{changes}{deletedmarkup '#1' unknown, using '\Changes@optiondeletedmarkup'}}
106 }
107 \typeout{changes-option 'deletedmarkup=\Changes@optiondeletedmarkup'}
108 }
```

Option highlightmarkup, stored or set to default value *background*.

```
109 \DeclareOptionX{highlightmarkup}{
110 \ifblank{#1}
111 {}
112 {
113 \IfIsInList{#1}{background|uuline|uwave}
114 {\renewcommand{\Changes@optionhighlightmarkup}{#1}}
115 {\PackageWarning{changes}{highlightmarkup '#1' unknown, using '\Changes@optionhighlightmarku
116 }
117 \typeout{changes-option 'highlightmarkup=\Changes@optionhighlightmarkup'}
```

118 *}*

Option `commentmarkup`, stored or set to default value *todo*.

119 *\DeclareOptionX{commentmarkup}{*
120 *\ifblank{#1}*
121 *{}*
122 *{*
123 *\IfIsInList{#1}{footnote|margin|todo|uwave}*
124 *{\renewcommand{\Changes@optioncommentmarkup}{#1}}*
125 *{\PackageWarning{changes}{commentmarkup '#1' unknown, using '\Changes@optioncommentmarkup'}}*
126 *}*
127 *\typeout{changes-option 'commentmarkup=\Changes@optioncommentmarkup'}*
128 *}*

Declare storage for authormarkup option and store option value or set to default value
superscript.

129 *\newcommand{\Changes@optionauthormarkup}{superscript}*
130 *\DeclareOptionX{authormarkup}{*
131 *\ifblank{#1}*
132 *{}*
133 *{*
134 *\IfIsInList{#1}{brackets|footnote|none|subscript|superscript}*
135 *{\renewcommand{\Changes@optionauthormarkup}{#1}}*
136 *{\PackageWarning{changes}{authormarkup '#1' unknown, using '\Changes@optionauthormarkup'}}*
137 *}*
138 *\typeout{changes-option 'authormarkup=\Changes@optionauthormarkup'}*
139 *}*

Declare storage for authormarkupposition option and store option value or set to default
value *right*.

140 *\newcommand{\Changes@optionauthormarkupposition}{right}*
141 *\DeclareOptionX{authormarkupposition}{*
142 *\ifblank{#1}*
143 *{}*
144 *{*
145 *\IfIsInList{#1}{left|right}*
146 *{\renewcommand{\Changes@optionauthormarkupposition}{#1}}*
147 *{\PackageWarning{changes}{authormarkupposition '#1' unknown, using '\Changes@optionauthormar*
148 *}*
149 *\typeout{changes-option 'authormarkupposition=\Changes@optionauthormarkupposition'}*
150 *}*

Declare storage for authormarkuptext option and store option value or set to default value
id.

151 *\newcommand{\Changes@optionauthormarkuptext}{id}*
152 *\DeclareOptionX{authormarkuptext}{*

```
153 \ifblank{#1}
154 {}
155 {
156 \IfIsInList{#1}{id|name}
157 {\renewcommand{\Changes@optionauthormarkuptext}{#1}}
158 {\PackageWarning{changes}{authormarkuptext '#1' unknown, using '\Changes@optionauthormarkupt
159 }
160 \typeout{changes-option 'authormarkuptext=\Changes@optionauthormarkuptext'}
161 }
```

Store default author color.

```
162 \newcommand{\Changes@optiondefaultcolor}{blue}
163 \DeclareOptionX{defaultcolor}{
164 \ifblank{#1}
165 {}
166 {
167 \renewcommand{\Changes@optiondefaultcolor}{#1}
168 }
169 \typeout{changes-option 'defaultcolor=\Changes@optiondefaultcolor'}
170 }
```

Options for package *todonotes* are directly passed to the package.

```
171 \DeclareOptionX{todonotes}{
172 \typeout{todonotes-option '#1', passed to package todonotes}
173 \PassOptionsToPackage{#1}{todonotes}
174 }
```

Options for package *truncate* are directly passed to the package.

```
175 \DeclareOptionX{truncate}{
176 \typeout{truncate-option '#1', passed to package truncate}
177 \PassOptionsToPackage{#1}{truncate}
178 }
```

Options for package *ulem* are directly passed to the package.

```
179 \DeclareOptionX{ulem}{
180 \typeout{ulem-option '#1', passed to package ulem}
181 \PassOptionsToPackage{#1}{ulem}
182 }
```

Options for package *xcolor* are directly passed to the package.

```
183 \DeclareOptionX{xcolor}{
184 \typeout{xcolor-option '#1', passed to package xcolor}
185 \PassOptionsToPackage{#1}{xcolor}
186 }
```

Unknown options generate a package warning.

```
187 \DeclareOptionX*{
188 \PackageWarning{changes}{Unknown option '\CurrentOption'}
189 }
```

10.1.2 Command options

All options for commands (e.g. \definechangesauthor) have to be declared before option processing.

\definechangesauthor

Declare available options of the command, define value storage.

```
190 \DeclareOptionX<Changes@definechangesauthor>{name}{\def\Changes@definechangesauthor@name{#1}
191 \DeclareOptionX<Changes@definechangesauthor>{color}{\def\Changes@definechangesauthor@color{#
```

Set the default values of the options.

```
192 \presetkeys{Changes@definechangesauthor}{
193 name=\@empty,
194 color=\Changes@optiondefaultcolor
195 }{}
```

\added

Declare available options of the command, define value storage.

```
196 \DeclareOptionX<Changes@added>{id}{\def\Changes@added@id{#1}}
197 \DeclareOptionX<Changes@added>{comment}{\def\Changes@added@comment{#1}}
```

Set the default values of the options.

```
198 \presetkeys{Changes@added}{
199 id=\@empty,
200 comment=\@empty,
201 }{}
```

`\deleted`

Declare available options of the command, define value storage.

```
202 \DeclareOptionX<Changes@deleted>{id}{\def\Changes@deleted@id{#1}}
203 \DeclareOptionX<Changes@deleted>{comment}{\def\Changes@deleted@comment{#1}}
```

Set the default values of the options.

```
204 \presetkeys{Changes@deleted}{
205 id=\@empty,
206 comment=\@empty,
207 }{}
```

`\replaced`

Declare available options of the command, define value storage.

```
208 \DeclareOptionX<Changes@replaced>{id}{\def\Changes@replaced@id{#1}}
209 \DeclareOptionX<Changes@replaced>{comment}{\def\Changes@replaced@comment{#1}}
```

Set the default values of the options.

```
210 \presetkeys{Changes@replaced}{
211 id=\@empty,
212 comment=\@empty,
213 }{}
```

`\highlight`

Declare available options of the command, define value storage.

```
214 \DeclareOptionX<Changes@highlight>{id}{\def\Changes@highlight@id{#1}}
215 \DeclareOptionX<Changes@highlight>{comment}{\def\Changes@highlight@comment{#1}}
```

Set the default values of the options.

```
216 \presetkeys{Changes@highlight}{
217 id=\@empty,
218 comment=\@empty,
219 }{}
```

`\comment`

Declare available options of the command, define value storage.

```
220 \DeclareOptionX<Changes@comment>{id}{\def\Changes@comment@id{#1}}
```

Set the default values of the options.

```
221 \presetkeys{Changes@comment}{
222 id=\@empty,
223 }{}
```

`\listofchanges`

Declare available options of the command, define value storage.

```
224 \DeclareOptionX<Changes@loc>{style}{\def\Changes@loc@style{#1}}
225 \DeclareOptionX<Changes@loc>{title}{\def\Changes@loc@title{#1}}
226 \DeclareOptionX<Changes@loc>{show}{\def\Changes@loc@show{#1}}
```

Set the default values of the options.

```
227 \presetkeys{Changes@loc}{
228 style=list,
229 title=\@empty,
230 show=all,
231 }{}
```

10.1.3 Package options

In order to avoid option clashes for options, state them here instead at the moment of requiring the package. Thanks for Markus Pahlow for pointing this out and providing the solution.

```
232 \ExecuteOptionsX{
233 ulem={normalem,normalbf},
234 truncate={breakall,fit}
235 }
```

10.1.4 Option processing

Process the options.

```
236 \ProcessOptionsX*\relax
```

10.2 Utility tests

\IfIsColored   Check if text should be colored.

Corresponds to the if macros of *etoolbox*, so it can be used in it's boolean tests. Mainly the last two parameters are the results of *true* and *false* computation.

```
237 \newrobustcmd{\IfIsColored}[2]{%
238 \IfStrEq{\Changes@optionmarkup}{nocolor}%
239 {#2}%
240 {#1}%
241 }
```

\IfIsEmpty   Checks if text in #1 is empty, executes #2 if empty, #3 otherwise.

Corresponds to the if macros of *etoolbox*, so it can be used in it's boolean tests. Mainly the last two parameters are the results of *true* and *false* computation.

I cannot add the test with \IfStrEq to \IfIsEmpty because it breaks e.g. with a \cite command in the final option, see issue #102. This test does not work (yet) with \csuse as parameter, see \Changes@output@author for the problem.

```
242 \newrobustcmd{\IfIsEmpty}[3]{%
243 \ifboolexpr{%
244 test {\ifblank{#1}} or%
245 test {\ifstrempty{#1}} or%
246 test {\ifdefstring{#1}{}} or%
247 test {\ifdefstring{#1}{\@empty}}%
248 }%
249 {#2}%
250 {#3}%
251 }
```

\IfIsAnonymous   Check if author id is empty, therefore the author is anonymous.

Corresponds to the if macros of *etoolbox*, so it can be used in it's boolean tests. Mainly the last two parameters are the results of *true* and *false* computation.

```
252 \newrobustcmd{\IfIsAnonymous}[3]{%
253 \IfIsEmpty{#1}{#2}{#3}%
254 }
```

EmptyAtPosition   Check if author is anonymous or position does not equal needed position, therefore the author text is empty.

This test could be removed if the test for empty \Changes@output@author would work. todo

Corresponds to the if macros of *etoolbox*, so it can be used in it's boolean tests. Mainly the last two parameters are the results of *true* and *false* computation.

```
255 \newrobustcmd{\IfIsAuthorEmptyAtPosition}[4]{%
256 \ifboolexpr{%
257 test {\IfIsAuthorOutputEmpty{#1}} or%
258 not test {\IfStrEq{\Changes@optionauthormarkupposition}{#2}}%
259 }%
260 {#3}%
261 {#4}%
262 }
```

uthorOutputEmpty Check if author output is empty.

This test could be removed if the test for empty \Changes@output@author would work. todo

Corresponds to the if macros of *etoolbox*, so it can be used in it's boolean tests. Mainly the last two parameters are the results of *true* and *false* computation.

```
263 \newrobustcmd{\IfIsAuthorOutputEmpty}[3]{%
264 \ifboolexpr{%
265 test {\IfIsAnonymous{#1}} and%
266 not test {\IfStrEq{\Changes@optionauthormarkuptext}{name}}%
267 }%
268 {#2}%
269 {#3}%
270 }
```

10.3 Packages

Package *xcolor* provides colored text.

```
271 \IfIsColored
272 {\RequirePackage{xcolor}}
273 {}
```

Package *ulem* provides commands for striking out text. Providing the needed package options via \ExecuteOptionsX.

```
274 \ifboolexpr{
275 test {\IfIsInList{\Changes@optionaddedmarkup}{dashuline|dotuline|sout|uline|uuline|uwave|xou
276 test {\IfIsInList{\Changes@optiondeletedmarkup}{dashuline|dotuline|sout|uline|uuline|uwave|x
277 test {\IfIsInList{\Changes@optionhighlightmarkup}{dashuline|dotuline|sout|uline|uuline|uwave
278 }
279 {\RequirePackage{ulem}}
280 {}
```

Package *todonotes* provides commands for todo notes in the margin.

```
281 \IfStrEq{\Changes@optioncommentmarkup}{todo}
282 {\RequirePackage{todonotes}}
283 {}
```

10.4 Language dependent texts

If the *babel* package is not loaded, the default language is English, in order to use another language, the user has to redefine the variables. If the *babel* or the *polyglossia* package is loaded, the default language is English too for undefined languages.

```
284 \def\listofchangesname{List of changes}
285 \def\summaryofchangesname{Changes}
286 \def\compactsummaryofchangesname{Changes (compact)}
287 \def\changesaddedname{Added}
288 \def\changesdeletedname{Deleted}
289 \def\changesreplacedname{Replaced}
290 \def\changeshighlightname{Highlighted}
291 \def\changescommentname{Commented}
292 \def\changesauthorname{Author}
293 \def\changesanonymousname{anonymous}
294 \def\changesnochanges{No changes.}
295 \def\changesnoloc{List of changes is available after the next \LaTeX\ run.}
296 \def\changesnosoc{Summary of changes is available after the next \LaTeX\ run.}
```

The check for *babel* or *polyglossia*, define language dependent texts afterwards.

```
297 \ifboolexpr{
298 test {\@ifpackageloaded{babel}} or
299 test {\@ifpackageloaded{polyglossia}}
300 }
301 {
302 \addto\captionsngerman{\def\listofchangesname{Liste der \"Anderungen}}
303 \addto\captionsngerman{\def\summaryofchangesname{\"Anderungen}}
304 \addto\captionsngerman{\def\compactsummaryofchangesname{\"Anderungen (kompakt)}}
305 \addto\captionsngerman{\def\changesaddedname{Eingef\"ugt}}
306 \addto\captionsngerman{\def\changesdeletedname{Gel\"oscht}}
307 \addto\captionsngerman{\def\changesreplacedname{Ersetzt}}
308 \addto\captionsngerman{\def\changeshighlightname{Hervorgehoben}}
309 \addto\captionsngerman{\def\changescommentname{Kommentiert}}
310 \addto\captionsngerman{\def\changesauthorname{Autor:in}}
311 \addto\captionsngerman{\def\changesanonymousname{Anonym}}
312 \addto\captionsngerman{\def\changesnochanges{Keine \"Anderungen.}}
313 \addto\captionsngerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \La-
    TeX-Lauf verf\"ugbar.}}
314 \addto\captionsngerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \La-
    TeX-Lauf verf\"ugbar.}}
```

```
315
316 \addto\captionsgerman{\def\listofchangesname{Liste der \"Anderungen}}
317 \addto\captionsgerman{\def\summaryofchangesname{\"Anderungen}}
318 \addto\captionsgerman{\def\compactsummaryofchangesname{\"Anderungen (kompakt)}}
319 \addto\captionsgerman{\def\changesaddedname{Eingef\"ugt}}
320 \addto\captionsgerman{\def\changesdeletedname{Gel\"oscht}}
321 \addto\captionsgerman{\def\changesreplacedname{Ersetzt}}
322 \addto\captionsgerman{\def\changeshighlightname{Hervorgehoben}}
323 \addto\captionsgerman{\def\changescommentname{Kommentiert}}
324 \addto\captionsgerman{\def\changesauthorname{Autor:in}}
325 \addto\captionsgerman{\def\changesanonymousname{Anonym}}
326 \addto\captionsgerman{\def\changesnochanges{Keine \"Anderungen.}}
327 \addto\captionsgerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \La-
    TeX-Lauf verf\"ugbar.}}
328 \addto\captionsgerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \La-
    TeX-Lauf verf\"ugbar.}}
329
330 \addto\captionsenglish{\def\listofchangesname{List of changes}}
331 \addto\captionsenglish{\def\summaryofchangesname{Changes}}
332 \addto\captionsenglish{\def\compactsummaryofchangesname{Changes (compact)}}
333 \addto\captionsenglish{\def\changesaddedname{Added}}
334 \addto\captionsenglish{\def\changesdeletedname{Deleted}}
335 \addto\captionsenglish{\def\changesreplacedname{Replaced}}
336 \addto\captionsenglish{\def\changeshighlightname{Highlighted}}
337 \addto\captionsenglish{\def\changescommentname{Commented}}
338 \addto\captionsenglish{\def\changesauthorname{Author}}
339 \addto\captionsenglish{\def\changesanonymousname{anonymous}}
340 \addto\captionsenglish{\def\changesnochanges{No changes.}}
341 \addto\captionsenglish{\def\changesnoloc{List of changes is available after the next \La-
    TeX\ run.}}
342 \addto\captionsenglish{\def\changesnosoc{Summary of changes is available af-
    ter the next \LaTeX\ run.}}
343
344 \addto\captionsbritish{\def\listofchangesname{List of changes}}
345 \addto\captionsbritish{\def\summaryofchangesname{Changes}}
346 \addto\captionsbritish{\def\compactsummaryofchangesname{Changes (compact)}}
347 \addto\captionsbritish{\def\changesaddedname{Added}}
348 \addto\captionsbritish{\def\changesdeletedname{Deleted}}
349 \addto\captionsbritish{\def\changesreplacedname{Replaced}}
350 \addto\captionsbritish{\def\changeshighlightname{Highlighted}}
351 \addto\captionsbritish{\def\changescommentname{Commented}}
352 \addto\captionsbritish{\def\changesauthorname{Author}}
353 \addto\captionsbritish{\def\changesanonymousname{anonymous}}
354 \addto\captionsbritish{\def\changesnochanges{No changes.}}
355 \addto\captionsbritish{\def\changesnoloc{List of changes is available after the next \La-
    TeX\ run.}}
356 \addto\captionsbritish{\def\changesnosoc{Summary of changes is available af-
    ter the next \LaTeX\ run.}}
357
```

```
358 \addto\captionsitalian{\def\listofchangesname{Lista delle modifiche}}
359 \addto\captionsitalian{\def\summaryofchangesname{Modifiche}}
360 \addto\captionsitalian{\def\compactsummaryofchangesname{Modifiche (coerente)}} % trans-
    lation by me (EK), please provide correct translation
361 \addto\captionsitalian{\def\changesaddedname{Aggiunte}}
362 \addto\captionsitalian{\def\changesdeletedname{Cancellazioni}}
363 \addto\captionsitalian{\def\changesreplacedname{Sostituzioni}}
364 \addto\captionsitalian{\def\changeshighlightname{Accentare}} % translation by me (EK), pleas
    vide correct translation
365 \addto\captionsitalian{\def\changescommentname{Commenti}} % translation by me (EK), please p
    vide correct translation
366 \addto\captionsitalian{\def\changesauthorname{Autore}}
367 \addto\captionsitalian{\def\changesanonymousname{anonimo}}
368 \addto\captionsitalian{\def\changesnochanges{Nessuna modifica.}} % translation by me (EK), p
    vide correct translation
369 \addto\captionsitalian{\def\changesnoloc{La lista delle modifiche sar\'a disponi-
    bile alla prossima esecuzione di \LaTeX.}}
370 \addto\captionsitalian{\def\changesnosoc{Le modifiche sar\'a disponibile alla prossima es-
    ecuzione di \LaTeX.}}
371
372 \addto\captionsfrench{\def\listofchangesname{Liste des modifications}}
373 \addto\captionsfrench{\def\summaryofchangesname{Modifications}}
374 \addto\captionsfrench{\def\compactsummaryofchangesname{Modifications (compact)}}
375 \addto\captionsfrench{\def\changesaddedname{Ajout\'e}}
376 \addto\captionsfrench{\def\changesdeletedname{Supprim\'e}}
377 \addto\captionsfrench{\def\changesreplacedname{Remplac\'e}}
378 \addto\captionsfrench{\def\changeshighlightname{Mis en \'evidence}}
379 \addto\captionsfrench{\def\changescommentname{Comment\'e}}
380 \addto\captionsfrench{\def\changesauthorname{Auteur}}
381 \addto\captionsfrench{\def\changesanonymousname{Anonyme}}
382 \addto\captionsfrench{\def\changesnochanges{Pas de modification}}
383 \addto\captionsfrench{\def\changesnoloc{Liste des modifications disponible apr\'es la procha
    TeX.}}
384 \addto\captionsfrench{\def\changesnosoc{Résumé des modifications disponible apr\'es la proch
    TeX.}}
385 }
386 {}
```

10.5 File extensions

es@locextension   Store file extension for list of changes, set default to *loc*.

```
387 \newcommand{\Changes@locextension}{loc}
```

setlocextension   Sets a new file extension for list of changes.

```
388 \newcommand{\setlocextension}[1]{
389 \renewcommand{\Changes@locextension}{#1}
```

390 *}*

es@socextension   Store file extension for summary of changes, set default to *soc*.

391 *\newcommand{\Changes@socextension}{soc}*

setsocextension   Sets a new file extension for summary of changes.

392 *\newcommand{\setsocextension}[1]{*
393 *\renewcommand{\Changes@socextension}{#1}*
394 *}*

10.6 Authors

10.6.1 Author management

Author counter.

395 *\newcounter{Changes@AuthorCount}*
396 *\setcounter{Changes@AuthorCount}{0}*
397 *\newcounter{Changes@Author}*

nechangesauthor   Define a new author. Mandatory argument: author's id. Optional arguments (key-value): author's name (default: empty) and author's color (default: \Changes@optiondefaultcolor (blue)).

Store id, name and color using named variables. Define counter and color per author.

398 *\newcommand*\definechangesauthor[2][]{*

Call *setkeys* in order to evaluate the key-value-options and fill the value storage.

399 *\setkeys{Changes@definechangesauthor}{#1}*

Increment author counter, later needed for *while* loop of authors.

400 *\stepcounter{Changes@AuthorCount}*

Store the id in a name with the given counter/index. All other storage refers to the id.

401 *\@namedef{Changes@AuthorID\theChanges@AuthorCount}{#2}*

Store the author's definition in according variables/colors, create change counters.

```
402 \expandafter\let\csname Changes@AuthorName#2\endcsname=\Changes@definechangesauthor@name
403 \expandafter\let\csname Changes@AuthorColor#2\endcsname=\Changes@definechangesauthor@color
404 \newcounter{Changes@addedCount#2}
405 \newcounter{Changes@deletedCount#2}
406 \newcounter{Changes@replacedCount#2}
407 \newcounter{Changes@highlightCount#2}
408 \newcounter{Changes@commentCount#2}
409 }
```

Define default-author (anonymous) with empty id and default color.

```
410 \definechangesauthor{\@empty}
```

setanonymousname   Sets the anonymous author name.

```
411 \newrobustcmd{\setanonymousname}[1]{
412 \AtBeginDocument{
413 \def\changesanonymousname{#1}
414 }
415 }
```

10.6.2 Author markup

Changes@Markup@author   Store markup for authors.

```
416 \newcommand{\Changes@Markup@author}[1]{%
417 \IfStrEq{\Changes@optionauthormarkup}{superscript}%
418 {\textsuperscript{#1}}%
419 {}%
420 \IfStrEq{\Changes@optionauthormarkup}{subscript}%
421 {\textsubscript{#1}}%
422 {}%
423 \IfStrEq{\Changes@optionauthormarkup}{brackets}%
424 {(#1)}%
425 {}%
426 \IfStrEq{\Changes@optionauthormarkup}{footnote}%
427 {\footnote{#1}}%
428 {}%
429 \IfStrEq{\Changes@optionauthormarkup}{none}%
430 {}%
431 {}%
432 }
```

setauthormarkup   Set markup for authors.

```
433 \newcommand{\setauthormarkup}[1]{
434 \renewcommand{\Changes@Markup@author}[1]{#1}
435 }
```

rmarkupposition   Set position for author markup text.

```
436 \newcommand{\setauthormarkupposition}[1]{
437 \renewcommand{\Changes@optionauthormarkupposition}{#1}
438 }
```

uthormarkuptext   Set author markup text to be displayed.

```
439 \newcommand{\setauthormarkuptext}[1]{
440 \renewcommand{\Changes@optionauthormarkuptext}{#1}
441 }
```

10.7 Change management commands

10.7.1 Text markup definition

Replaced text is always typeset as follows: ⟨*added text*⟩⟨*deleted text*⟩. Therefore no extra command for markup of replaced text is given.

\Changes@Markup   Predefined markups for text.

```
442 \newcommand{\Changes@Markup}[2]{%
443 \IfStrEq{#1}{background}%
444 {%
445 \IfIsColored%
446 {\colorbox{authorcolor!30}{#2}}%
447 {#2}%
448 }%
449 {}%
450 \IfStrEq{#1}{bf}%
451 {\textbf{#2}}%
452 {}%
453 \IfStrEq{#1}{colored}%
454 {#2}%
455 {}%
456 \IfStrEq{#1}{dashuline}%
457 {\dashuline{#2}}%
458 {}%
459 \IfStrEq{#1}{dotuline}%
460 {\dotuline{#2}}%
461 {}%
```

```
462 \IfStrEq{#1}{em}%
463 {\emph{#2}}%
464 {}%
465 \IfStrEq{#1}{it}%
466 {\textit{#2}}%
467 {}%
468 \IfStrEq{#1}{sl}%
469 {\textsl{#2}}%
470 {}%
471 \IfStrEq{#1}{sout}%
472 {\sout{#2}}%
473 {}%
474 \IfStrEq{#1}{uline}%
475 {\uline{#2}}%
476 {}%
477 \IfStrEq{#1}{uuline}%
478 {\uuline{#2}}%
479 {}%
480 \IfStrEq{#1}{uwave}%
481 {\uwave{#2}}%
482 {}%
483 \IfStrEq{#1}{xout}%
484 {\xout{#2}}%
485 {}%
486 }
```

es@Markup@added   Store markup for added text.

```
487 \newcommand{\Changes@Markup@added}[1]{%
488 \Changes@Markup{\Changes@optionaddedmarkup}{#1}{}%
489 }
```

\setaddedmarkup   Set markup for added text.

```
490 \newcommand{\setaddedmarkup}[1]{
491 \renewcommand{\Changes@Markup@added}[1]{#1}
492 }
```

@Markup@deleted   Store markup for deleted text.

```
493 \newcommand{\Changes@Markup@deleted}[1]{%
494 \Changes@Markup{\Changes@optiondeletedmarkup}{#1}{}%
495 }
```

etdeletedmarkup   Set markup for deleted text.

```
496 \newcommand{\setdeletedmarkup}[1]{
497 \renewcommand{\Changes@Markup@deleted}[1]{#1}
498 }
```

arkup@highlight   Store markup for highlighted text.

```
499 \newcommand{\Changes@Markup@highlight}[1]{%
500 \Changes@Markup{\Changes@optionhighlightmarkup}{#1}{}%
501 }
```

highlightmarkup   Set markup for highlighted text.

```
502 \newcommand{\sethighlightmarkup}[1]{
503 \renewcommand{\Changes@Markup@highlight}[1]{#1}
504 }
```

@Markup@comment   Store markup for comments.

Parameters:

1. text
2. author's id
3. author's id/name output

```
505 \newcommand{\Changes@Markup@comment}[3]{%
506 \IfStrEq{\Changes@optioncommentmarkup}{todo}%
507 {%
508 \IfIsColored%
509 {\colorlet{Changes@todocolor}{authorcolor}}%
510 {\colorlet{Changes@todocolor}{black}}%
511 \todo[color=Changes@todocolor!10, bordercolor=Changes@todocolor, linecolor=Changes@todocolor
    list]{%
512 \textbf{[\IfIsAuthorOutputEmpty{#2}{}{#3~}\arabic{authorcommentcount}]} #1%
513 }%
514 }{}%
515 \IfStrEq{\Changes@optioncommentmarkup}{margin}%
516 {%
517 \marginpar{%
518 \IfIsColored%
519 {\leavevmode\color{authorcolor}}%
520 {}%
521 \textbf{[\IfIsAuthorOutputEmpty{#2}{}{#3~}\arabic{authorcommentcount}]:} #1%
522 }%
523 }{}%
524 \IfStrEq{\Changes@optioncommentmarkup}{footnote}%
525 {%
526 \footnote{%
527 \textbf{[\IfIsAuthorOutputEmpty{#2}{}{#3~}\arabic{authorcommentcount}]:} #1%
528 }%
529 }{}%
530 \IfStrEq{\Changes@optioncommentmarkup}{uwave}%
531 {%
532 {%
```

```
533 \IfIsColored%
534 {\color{authorcolor}}%
535 {}%
536 \allowbreak%
537 \uwave{%
538 \textbf{[\IfIsAuthorOutputEmpty{#2}{}{#3~}\arabic{authorcommentcount}]:} #1%
539 }%
540 }%
541 }{}%
542 }
```

etcommentmarkup Set markup for comments.

```
543 \newcommand{\setcommentmarkup}[1]{
544 \renewcommand{\Changes@Markup@comment}[3]{#1}
545 }
```

10.7.2 Change management command definition

s@check@author Check if author id is valid. An empty id is valid by default.

If the id is not valid, a package error is raised.

Loop code is taken from `https://texfaq.org/FAQ-repeat-num`

This command has the following arguments:

1. author's id

```
546 \newbool{Changes@WrongID}
547 \newcommand{\Changes@check@author}[1]{%
548 \IfIsEmpty{#1}%
549 {}%
550 {%
551 \setbool{Changes@WrongID}{true}%
552 \setcounter{Changes@Author}{0}%
553 \loop%
554 \stepcounter{Changes@Author}%
555 \IfStrEq{#1}{\csuse{Changes@AuthorID\theChanges@Author}}%
556 {\setbool{Changes@WrongID}{false}}%
557 {}%
558 \ifnum\theChanges@Author<\theChanges@AuthorCount%
559 \repeat%
560 \ifbool{Changes@WrongID}%
561 {%
562 \PackageError{changes}%
563 {Undefined changes author: #1}%
564 {You have to define the author #1 with e.g.: \definechangesauthor{#1}}%
565 }%
```

```
566 {}%
567 }%
568 }
```

s@output@author Output command for the author: id or name.

This command has the following arguments:

1. author's id

```
569 \newrobustcmd{\Changes@output@author}[1]{%
570 \IfStrEq{\Changes@optionauthormarkuptext}{id}%
571 {#1}%
572 {}%
573 \IfStrEq{\Changes@optionauthormarkuptext}{name}%
574 {%
575 \IfIsAnonymous{#1}%
576 {\changesanonymousname}%
577 {%
```

I cannot use \IfIsEmpty here, because it does not work in this case. I cannot add the test with \IfStrEq to \IfIsEmpty because it breaks e.g. with a \cite command in the final option.

```
578 \IfStrEq{\csuse{Changes@AuthorName#1}}{}%
579 {#1}%
580 {\csuse{Changes@AuthorName#1}}%
581 }%
582 }%
583 {}%
584 }
```

author@position Output command for the author at the given position.

This command has the following arguments:

1. author's id
2. position to output the author to (left or right)

\DeclareRobustCommand is used for not breaking the todo note definition.

```
585 \newrobustcmd{\Changes@output@author@position}[2]{%
586 \IfIsAuthorEmptyAtPosition{#1}{#2}%
587 {}%
588 {%
589 \Changes@output@author{#1}%
590 }%
591 }
```

anges@set@color   Sets the author's color.

This command has the following argument:

1. author's id

```
592 \newrobustcmd{\Changes@set@color}[1]{%
593 \IfIsColored%
594 {\colorlet{authorcolor}{\csuse{Changes@AuthorColor#1}}}%
595 {}%
596 }
```

et@commentcount   Sets the author's comment count.

This command has the following argument:

1. author's id

```
597 \newcounter{authorcommentcount}
598 \newrobustcmd{\Changes@set@commentcount}[1]{%
599 \setcounter{authorcommentcount}{\value{Changes@commentCount#1}}%
600 }
```

set@commandname   Sets the command name according to setting of commandname.

```
601 \def\Changes@commandprefix{ch}
602 \newrobustcmd{\Changes@set@commandname}[1]{
603 \def\Changes@commandname{#1}
604 \IfStrEq{\Changes@optioncommandnameprefix}{always}
605 {
606 \def\Changes@commandname{\Changes@commandprefix#1}
607 \typeout{changes-package option commandnameprefix=always: using prefix on #1, new com-
    mand name is \Changes@commandname.}
608 }{
609 \ifcsdef{#1}
610 {
611 \IfStrEq{\Changes@optioncommandnameprefix}{none}
612 {
613 \PackageError{changes}
614 {Command #1 is already defined.}
615 {Try package option commandnameprefix with "always" or "ifneeded", e.g. usep-
    ackage[commandnameprefix=always]{changes}}
616 }{}
617 \IfStrEq{\Changes@optioncommandnameprefix}{ifneeded}
618 {
619 \def\Changes@commandname{\Changes@commandprefix#1}
620 \PackageWarning{changes}{Command #1 is already defined, using \Changes@commandname}
621 }{}
622 }
623 {}
```

```
624 }
625 }
```

\Changes@output Output command for the changed text.

This command has the following arguments:

1. change id (added, deleted, replaced, highlight, comment)
2. author's id
3. final text
4. deleted/replaced text
5. comment
6. change type name for list of changes
7. text for list of changes

```
626 \newrobustcmd{\Changes@output}[7]{%
```

Output changed text if option draft is set, otherwise output unchanged text.

```
627 \ifbool{Changes@optiondraft}%
628 {%
```

Check if author's id is valid and set author's color.

```
629 \Changes@check@author{#2}%
630 \Changes@set@color{#2}%
```

Start output.

```
631 {%
```

Output for change commands: added, deleted, replaced, highlight.

I think this code is not elegant but it gets the work done for now.

```
632 \IfIsInList{#1}{added|deleted|replaced|highlight}%
633 {%
```

Author text for left positioning (only without comment).

```
634 \IfIsEmpty{#5}%
635 {%
636 \IfIsAuthorEmptyAtPosition{#2}{left}%
637 {}%
638 {{%
639 \IfIsColored%
640 {\color{authorcolor}}%
641 {}%
642 \Changes@Markup@author{\Changes@output@author@position{#2}{left}}%
643 }}%
644 }{}%
```

Changed/highlighted text.

```
645 {%
646 \IfStrEq{#1}{highlight}%
647 {}{%
648 \IfIsColored%
649 {\color{authorcolor}}%
650 {}%
651 }%
652 \IfStrEq{#1}{added}{\Changes@Markup@added{#3}}{}%
653 \IfStrEq{#1}{deleted}{\Changes@Markup@deleted{#4}}{}%
654 \IfStrEq{#1}{replaced}{{\Changes@Markup@added{#3}}\allowbreak\Changes@Markup@deleted{#4}}{}%
655 \IfStrEq{#1}{highlight}{\Changes@Markup@highlight{#3}}{}%
656 }%
```

Author text for right positioning (only without comment).

```
657 \IfIsEmpty{#5}%
658 {%
659 \IfIsAuthorEmptyAtPosition{#2}{right}%
660 {}%
661 {{%
662 \IfIsColored%
663 {\color{authorcolor}}%
664 {}%
665 \Changes@Markup@author{\Changes@output@author@position{#2}{right}}%
666 }}%
667 }{}%
```

Update counters.

```
668 \stepcounter{Changes@#1Count#2}%
669 }{}%
```

Comments.

```
670 \IfIsEmpty{#5}%
671 {}%
672 {%
673 \stepcounter{Changes@commentCount#2}%
674 \Changes@set@commentcount{#2}%
675 \Changes@Markup@comment%
676 {#5}%
677 {#2}%
678 {\Changes@output@author{#2}}%
679 }%
680 }%
```

Store line for list of changes.

```
681 \IfIsEmpty{#2}%
682 {\def\Changes@locid{}}%
683 {\def\Changes@locid{~(#2)}}%
684 \phantomsection\addcontentsline{\Changes@locextension}{#1}{#6\Changes@locid: \trun-
    cate{\Changestruncatewidth}{#7}}%
685 }%
```

Output unchanged text (option `final` was set).

```
686 {%
687 \IfIsEmpty{#3}%
688 {\@bsphack\@esphack}%
689 {#3}%
690 }%
691 }
```

\added   The command formats text as new text.

Mandatory argument: added text. Optional argument (key-value): author's id, comment

```
692 \Changes@set@commandname{added}
693 \expandafter\newcommand\expandafter{\csname\Changes@commandname\endcsname}[2][\@empty]{%
```

Call *setkeys* in order to evaluate the key-value-options and fill the value storage.

```
694 \setkeys{Changes@added}{#1}%
695 \Changes@output%
696 {added}%
697 {\Changes@added@id}%
698 {#2}%
699 {}%
700 {\Changes@added@comment}%
701 {\changesaddedname}%
702 {#2}%
703 }
```

\deleted   The command formats text as deleted text.

The definition of the empty text for unchanged text is provided by Frank Mittelbach, slightly modified by me. It solves the problem of additional space caused by an empty command (e.g. when using the `final` option). Before that, there was a slightly erroneous version from de.comp.text.tex (issue #2).

Mandatory argument: deleted text. Optional argument (key-value): author's id, comment

```
704 \Changes@set@commandname{deleted}
705 \expandafter\newcommand\expandafter{\csname\Changes@commandname\endcsname}[2][\@empty]{%
```

Call *setkeys* in order to evaluate the key-value-options and fill the value storage.

```
706 \setkeys{Changes@deleted}{#1}%
707 \Changes@output%
708 {deleted}%
709 {\Changes@deleted@id}%
710 {}%
711 {#2}%
712 {\Changes@deleted@comment}%
713 {\changesdeletedname}%
714 {#2}%
715 }
```

\replaced   The command formats text as replaced text.

Mandatory arguments: new text and old text. Optional argument (key-value): author's id, comment

```
716 \Changes@set@commandname{replaced}
717 \expandafter\newcommand\expandafter{\csname\Changes@commandname\endcsname}[3][\@empty]{%
```

Call *setkeys* in order to evaluate the key-value-options and fill the value storage.

```
718 \setkeys{Changes@replaced}{#1}%
719 \Changes@output%
720 {replaced}%
721 {\Changes@replaced@id}%
722 {#2}%
723 {#3}%
724 {\Changes@replaced@comment}%
725 {\changesreplacedname}%
726 {#2}%
727 }
```

\highlight   The command formats text as highlighted text.

Mandatory argument: highlighted text. Optional argument (key-value): author's id, comment

```
728 \Changes@set@commandname{highlight}
729 \expandafter\newcommand\expandafter{\csname\Changes@commandname\endcsname}[2][\@empty]{%
```

Call *setkeys* in order to evaluate the key-value-options and fill the value storage.

```
730 \setkeys{Changes@highlight}{#1}%
731 \Changes@output%
732 {highlight}%
733 {\Changes@highlight@id}%
734 {#2}%
735 {}%
```

```
736 {\Changes@highlight@comment}%
737 {\changeshighlightname}%
738 {#2}%
739 }
```

\comment The command formats text as comment.

Mandatory argument: comment text. Optional argument (key-value): author's id

```
740 \Changes@set@commandname{comment}
741 \expandafter\newcommand\expandafter{\csname\Changes@commandname\endcsname}[2][\@empty]{%
```

Call *setkeys* in order to evaluate the key-value-options and fill the value storage.

```
742 \setkeys{Changes@comment}{#1}%
743 \Changes@output%
744 {comment}%
745 {\Changes@comment@id}%
746 {}%
747 {}%
748 {#2}%
749 {\changescommentname}%
750 {#2}%
751 }
```

10.8 List of changes

The list of changes truncates text, therefore the *truncate* package is used. (Using fit and redefining the marker: suggestion and code by Frank Mittelbach) Providing the needed package options via \ExecuteOptionsX.

```
752 \RequirePackage{truncate}
753 \renewcommand\TruncateMarker{ [\dots\negthinspace]\ }
```

hanges@chopline Auxiliary command for reading the content of the loc-files. The content is read line by line. One line is parsed with this macro, the order of entries is: id, color, name, added, deleted, replaced, highlighted, comment. The contents have to be separated by a semicolon.

```
754 \def\changes@chopline#1;#2;#3;#4;#5;#6;#7;#8 \\{%
755 \def\Changes@Inid{#1}%
756 \def\Changes@Incolor{#2}%
757 \def\Changes@Inname{#3}%
758 \def\Changes@Inadded{#4}%
759 \def\Changes@Indeleted{#5}%
760 \def\Changes@Inreplaced{#6}%
761 \def\Changes@Inhighlight{#7}%
762 \def\Changes@Incomment{#8}%
763 }
```

ChangesListline   Output of a list line, calls \contentsline for output which uses \l@change.

This command has the following arguments:

1. change type (added, ...)
2. change description with author
3. page
4. page reference for hyperref

```
764 \newcommand{\ChangesListline}[4]{%
765 \IfIsInList{#1}{\Changes@loc@show}{%
766 \@ifundefined{@dotsep}%
767 {\def\@dotsep{4.5}}%
768 {}%
769 \origcontentsline{change}{#2}{#3}{#4}
770 }{}%
771 }
```

\l@change   Output of a loc entry.

```
772 \newcommand*\l@change{\@dottedtocline{1}{0pt}{2em}}
```

ettruncatewidth   Set the width of the truncation. Argument: new width.

```
773 \newcommand{\settruncatewidth}[1]{
774 \setlength{\Changestruncatewidth}{#1}
775 }
```

estruncatewidth   Length for the width of the truncation.

Default: two third of the text width

Normally, the name would be separated by @, as all internal macros are, but this leads to compile errors when processing the list of changes, therefore the @ were omitted.

```
776 \newlength{\Changestruncatewidth}
777 \settruncatewidth{.6\textwidth}
```

setsummarywidth   Set the width of the change summary. Argument: new width.

```
778 \newcommand{\setsummarywidth}[1]{
779 \setlength{\Changes@summary@width}{#1}
780 }
```

tsummarytowidth   Set the width of the change summary to width of given text. Argument: text.

```
781 \newcommand{\setsummarytowidth}[1]{
782 \settowidth{\Changes@summary@width}{#1}
783 }
```

s@summary@width Length for the width of the change summary.

Default: one third of the text width

```
784 \newlength{\Changes@summary@width}
785 \setsummarywidth{.3\textwidth}
```

\phantomsection Provide command in case *hyperref* is not loaded.

```
786 \providecommand\phantomsection{}
```

\listofchanges This command outputs the list of changes. Options: title, style and show.

The following styles are available:

list   prints the list of changes like a list of figures
summary   prints the number of changes grouped by author
compactsummary   same as summary but entries with count 0 are omitted

The following show values are available:

all   all kinds of changes
added   print only additions (same for delete, replace, highlight, comment)
added|deleted   print additions and deletions (and other combinations with |)

For the lists, the values are read from the auxiliary files (loc and soc). If no loc/soc-files exists, an according message is generated.

```
787 \newcommand{\listofchanges}[1][\@empty]{%
788 \setkeys{Changes@loc}{#1}%
```

All work is done only in draft mode.

```
789 \ifbool{Changes@optiondraft}%
790 {%
```

Check if style is known, otherwise use list by default.

```
791 \IfIsInList{\Changes@loc@style}{list|summary|compactsummary}%
792 {}%
793 {%
794 \PackageWarning{changes}{Wrong style for list of changes: '\Changes@loc@style', us-
    ing 'list' instead.}%
795 \def\Changes@loc@style{}%
796 }%
797 \IfIsEmpty{\Changes@loc@style}%
798 {\def\Changes@loc@style{list}}%
799 {}%
```

Check if show-value is known, otherwise use `all` by default.

```
800 \IfStrEq{\Changes@loc@show}{all}%
801 {\def\Changes@loc@show{added|deleted|replaced|highlight|comment}}%
802 {}%
803 \IfIsInList{\Changes@loc@show}{added|deleted|replaced|highlight|comment}%
804 {}%
805 {%
806 \PackageWarning{changes}{Wrong show-value for list of changes: '\Changes@loc@show', us-
    ing 'all' instead.}%
807 \def\Changes@loc@show{}%
808 }%
809 \IfIsEmpty{\Changes@loc@show}%
810 {\def\Changes@loc@show{added|deleted|replaced|highlight|comment}}%
811 {}%
```

Print heading.

```
812 \IfIsEmpty{\Changes@loc@title}%
813 {%
814 \IfStrEq{\Changes@loc@style}{list}%
815 {\def\Changes@heading{\listofchangesname}}{}%
816 \IfStrEq{\Changes@loc@style}{summary}%
817 {\def\Changes@heading{\summaryofchangesname}}{}%
818 \IfStrEq{\Changes@loc@style}{compactsummary}%
819 {\def\Changes@heading{\compactsummaryofchangesname}}{}%
820 }%
821 {\def\Changes@heading{\Changes@loc@title}}%
822 \section*{\Changes@heading}
```

Print list.

The `\ifeof` command is neccessary in the loop too, otherwise there are problems with
the last (empty) line of the file.

The output is done with `\contentsline`, in order to filter the output I have to redefine
the command. I did not find a better solution than the `\let` command.

```
823 \IfIsInList{\Changes@loc@style}{list}%
824 {%
825 \IfFileExists{\jobname.\Changes@locextension}%
826 {%
827 \let\origcontentsline\contentsline%
828 \let\contentsline\ChangesListline%
829 \newread\Changes@InFile%
830 \openin\Changes@InFile=\jobname.\Changes@locextension%
831 \loop\unless\ifeof\Changes@InFile%
832 \read\Changes@InFile to \Changes@Line%
833 \ifeof\Changes@InFile\else%
834 \Changes@Line%
```

```
835 \fi
836 \repeat
837 \closein\Changes@InFile%
838 \let\contentsline\origcontentsline%
839 }{%
840 \emph{\changesnoloc}%
841 \PackageWarning{changes}{LaTeX rerun needed for list of changes}%
842 }%
843 }{}%
```

Print summary or compact summary.

The \ifeof command is neccessary in the loop too, otherwise there are problems with the last (empty) line of the file.

```
844 \IfIsInList{\Changes@loc@style}{summary|compactsummary}%
845 {%
846 \IfFileExists{\jobname.\Changes@socextension}%
847 {%
```

Deactivate semicolon as shorthand in order to avoid problems reading the semicolon separated lines.

```
848 \ifdef{\ifbabelshorthand}{%
849 \ifbabelshorthand{;}{\shorthandoff{;}}{}}{}%
850 }{}%
851 \newread\Changes@InFile%
852 \openin\Changes@InFile = \jobname.\Changes@socextension%
```

Every line contains the author's id, color and number of changes.

```
853 \loop\unless\ifeof\Changes@InFile%
854 \read\Changes@InFile to \Changes@Line%
855 \ifeof\Changes@InFile\else%
856 \expandafter\changes@chopline\Changes@Line\\%
857 \textbf{%
858 \IfIsColored%
859 {\color{\Changes@Incolor}}%
860 {}%
861 \IfIsAnonymous{\Changes@Inid}%
862 {\changesauthorname: \changesanonymousname}%
863 {%
864 \changesauthorname: \Changes@Inid%
865 \IfIsEmpty{\Changes@Inname}%
866 {}%
867 { (\Changes@Inname)}%
868 }%
869 }\\%
```

Compute the relevant sum of changes in order to print "no changes" correctly.

To make sure \do can be renewed, we define it first, some macros, such as \chapter undefine it.

```
870 \numdef{\Changes@InSum}{0}%
871 \def\do{}%
872 \renewcommand*{\do}[1]{%
873 \numdef{\Changes@InSum}{\Changes@InSum + \csuse{Changes@In########1}}%
874 }%
875 \expandafter\dopsvlist\expandafter{\Changes@loc@show}%
```

Print summary.

```
876 \ifnumcomp{\Changes@InSum}{=}{0}%
877 {%
878 \parbox{\Changes@summary@width}{\changesnochanges}\\[1ex]%
879 }%
880 {%
881 \numdef{\Changes@InCount}{0}%
882 \renewcommand*{\do}[1]{%
883 \numdef{\Changes@InCount}{\Changes@InCount + \csuse{Changes@In########1}}%
884 \ifboolexpr{%
885 not test {\IfStrEq{\Changes@loc@style}{compactsummary}} or%
886 test {\ifnumgreater{\csuse{Changes@In########1}}{0}}%
887 }%
888 {%
889 \parbox{\Changes@summary@width}{\csuse{changes########1name}~\let\cleaders\leaders\dotfill~\
890 \ifnumless{\Changes@InCount}{\Changes@InSum}%
891 {\\}%
892 {\\[1ex]}%
893 }%
894 {}%
895 }%
896 \expandafter\dopsvlist\expandafter{\Changes@loc@show}%
897 }%
898 \fi%
```

File read loop.

```
899 \repeat
900 \closein\Changes@InFile%
```

Reactivate semicolon as shorthand.

```
901 \ifdef{\ifbabelshorthand}{%
902 \ifbabelshorthand{;}{\shorthandon{;}}{}%
903 }{}%
904 }{%
905 \emph{\changesnosoc}%
```

```
906 \PackageWarning{changes}{LaTeX rerun needed for summary of changes}%
907 }%
908 }{}%
```

In final mode print nothing.

```
909 }{}%
910 }
```

At the end of the document: write the list of changes in the loc-file, therefore open file, write values, close file. Changes are written as LaTeX-formatted text, so they can simply be read via \input.

The order of entries is: id, color, name, added, deleted, replaced, comment, highlight. The contents have to be separated by a semicolon.

```
911 \AtEndDocument{%
```

Open output file.

```
912 \newwrite\Changes@OutFile
913 \immediate\openout\Changes@OutFile = \jobname.\Changes@socextension
```

Redefine expandable of \protect in order to write correct special characters. Store original definition for resetting \protect.

```
914 \let\Changes@protect\protect
915 \let\protect\@unexpandable@protect
```

Output data for list of changes.

```
916 \setcounter{Changes@Author}{0}
917 \loop\unless\ifnum\theChanges@Author = \value{Changes@AuthorCount}
918 \stepcounter{Changes@Author}%
919 \def\Changes@ID{\csuse{Changes@AuthorID\theChanges@Author}}%
920 \immediate\write\Changes@OutFile{\Changes@ID;%
921 \csuse{Changes@AuthorColor\Changes@ID};%
922 \csuse{Changes@AuthorName\Changes@ID};%
923 \the\value{Changes@addedCount\Changes@ID};%
924 \the\value{Changes@deletedCount\Changes@ID};%
925 \the\value{Changes@replacedCount\Changes@ID};%
926 \the\value{Changes@highlightCount\Changes@ID};%
927 \the\value{Changes@commentCount\Changes@ID}}%
928 \repeat
```

Close output file.

```
929 \immediate\closeout\Changes@OutFile
```

Restore original definition of \protect.

```
930 \let\protect\Changes@protect
```

Write content of listofchanges to file.

```
931 \if@filesw
932 \@ifundefined{tf@\Changes@locextension}{
933 \expandafter\newwrite\csname tf@\Changes@locextension\endcsname
934 \immediate\openout \csname tf@\Changes@locextension\endcsname \jobname.\Changes@locextension
935 }{}
936 \fi
937 }
```

```
938 ⟨/changes⟩
```