The *changes*-package

Manual change markup — version 3.1.3

July 21, 2019

Ekkart Kleinod

✉ ekleinod@edgesoft.de

1 Introduction **4**

2 Using the *changes*-package **5**

3 Limitations and possible enhancements **8**

4 User interface of the *changes*-package **9**

4.1 Package Options 9

4.1.1 draft 10

4.1.2 final 10

4.1.3 markup 10

4.1.4 addedmarkup 11

4.1.5 deletedmarkup 11

4.1.6 highlightmarkup 12

4.1.7 commentmarkup 12

4.1.8 authormarkup 13

4.1.9 authormarkupposition 14

4.1.10 authormarkuptext 14

4.1.11 todonotes 14

4.1.12 truncate 15

4.1.13 ulem 15

4.1.14 xcolor 15

4.2 Change management 15

4.2.1 `\added` 16

4.2.2 `\deleted` 16

4.2.3 `\replaced` 17

4.3 Highlighting and Comments 17

4.3.1 `\highlight` 17

4.3.2 `\comment` 18

4.4 Overview of changes 18

4.4.1 `\listofchanges` 18

4.5 Author management 19
4.5.1 \definechangesauthor 19
4.6 Adaptation of the output 20
4.6.1 \setaddedmarkup 21
4.6.2 \setdeletedmarkup 21
4.6.3 \sethighlightmarkup 22
4.6.4 \setcommentmarkup 22
4.6.5 \setauthormarkup 23
4.6.6 \setauthormarkupposition 23
4.6.7 \setauthormarkuptext 23
4.6.8 \settruncatewidth 24
4.6.9 \setsummarywidth 24
4.6.10 \setsummarytowidth 24
4.6.11 \setsocextension 25
4.7 Used packages 25

5 Known problems and solutions **26**
5.1 Special content 26
5.2 Footnotes and margin notes 26
5.3 The *ulem* package 26

6 Authors **28**

7 Versions **29**

8 Distribution, Copyright, License **30**

9 The documented sourcecode **31**
9.1 Package information and options 31
9.1.1 Package options 31
9.1.2 Command options 36

9.1.3 Package options 39
9.1.4 Option processing 39
9.2 Packages 39
9.3 Language dependent texts 40
9.4 File extension 42
9.5 Authors 42
9.5.1 Author management 42
9.5.2 Author markup 43
9.6 Change management commands 44
9.6.1 Text markup definition 44
9.6.2 Change management command definition 47
9.7 List of changes 56

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

`http://edgesoft.de/projects/latex/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 good~~bad~~[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 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
– problems with nested commands

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. In that case, please go to

`http://changes.sourceforge.net/`

There you find information on how to report errors or improvements, give advice to other users, or participate in the development of the package.

You can find a list of known problems and possible solutions in Section 5. Please refer to the section first if your problem is known and is a solution exists.

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
– 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.

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 10
4.1.2 final 10
4.1.3 markup 10
4.1.4 addedmarkup 11
4.1.5 deletedmarkup 11
4.1.6 highlightmarkup 12
4.1.7 commentmarkup 12
4.1.8 authormarkup 13
4.1.9 authormarkupposition 14
4.1.10 authormarkuptext 14
4.1.11 todonotes 14
4.1.12 truncate 15
4.1.13 ulem 15
4.1.14 xcolor 15

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 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.

4.1.4 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}
```

4.1.5 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 – 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[deletedmarkup=sout]{changes} ≙ \usepackage{changes}
\usepackage[deletedmarkup=xout]{changes}
\usepackage[deletedmarkup=uwave]{changes}
```

4.1.6 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 – example
`uwave`	wavy underlined text – example

Examples

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

4.1.7 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:

example comment

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}
```

4.1.8 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 – $\text{text}^{\text{author}}$ (default)
subscript	subscripted text – $\text{text}_{\text{author}}$
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

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

Examples

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

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

Examples

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

4.1.11 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}
```

4.1.12 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}
```

4.1.13 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}
```

4.1.14 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 16
4.2.2 \deleted 16
4.2.3 \replaced 17

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] anonymous

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

This is newreplaced text. This is newreplaced[EK] text too. This is more new replaced text. This is the last newreplaced text.

[EK 5] better

[3] improved

4.3 Highlighting and Comments

4.3.1 \highlight 17
4.3.2 \comment 18

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

This is commented text. This is commented text too.

[5] Sure

[EK 7] Correct.

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 \setaddedmarkup 21
4.6.2 \setdeletedmarkup 21
4.6.3 \sethighlightmarkup 22
4.6.4 \setcommentmarkup 22
4.6.5 \setauthormarkup 23
4.6.6 \setauthormarkupposition 23
4.6.7 \setauthormarkuptext 23
4.6.8 \settruncatewidth 24
4.6.9 \setsummarywidth 24
4.6.10 \setsummarytowidth 24
4.6.11 \setsocextension 25

4.6.1 \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:

– any LaTeX-commands
– 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}
```

4.6.2 \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:

– any LaTeX-commands
– 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}
```

4.6.3 \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:

– any LaTeX-commands
– highlighted text can be used with "#1"
– *ifthenelse* boolean test for colored text "\isColored"
– author's color can be used with color "authorcolor"

Examples

```
\sethighlightmarkup{\emph{#1}}
\sethighlightmarkup{\ifthenelse{\isColored}{\color{authorcolor
    }}{}\#\#: #1}
```

4.6.4 \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:

– any LaTeX-commands
– comment can be used with "#1"
– author's id can be used with "#2"
– author output (id or name) can be used with "#3"
– *ifthenelse* boolean test for anonymous author "\isAnonymous"
– *ifthenelse* boolean test for colored text "\isColored"
– author's color can be used with color "authorcolor"
– comment count of the autor can be used with counter "authorcommentcount"

Examples

```
\setcommentmarkup{-- #1 --}
\setcommentmarkup{\ifthenelse{\isColored}{\color{authorcolor}}{}#1}
\setcommentmarkup{\ifthenelse{\isAnonymous{#2}}{}{\textbf{#3: }}#1}
\setcommentmarkup{[\arabic{authorcommentcount}] #1}
```

4.6.5 \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:

– any LaTeX-commands
– author output (id or name) can be used with "#1"

Examples

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

4.6.6 \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}
```

4.6.7 \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}
```

4.6.8 \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}
```

4.6.9 \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}
```

4.6.10 \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 you can imagine for the summary.}
```

4.6.11 \setsocextension

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

The command \setsocextension sets the extension of the auxiliary file for the summary of changes (soc-file[3]). 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 "loc", as this would collide with the normal LaTeX run.

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:

xifthen	provides an enhanced \if-command as well as a while-loop
xkeyval	provides options with key-value-pairs
xstring	improves string operations

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

pdfcolmk	loaded if colored text is used for markup (default markup); solves the problem of colored text and page breaks (with pdflatex)
todonotes	loaded if comments are layouted as todo notes (default markup)
ulem	loaded if text has to be striked or exed out (default markup)
xcolor	loaded if colored text is used for markup (default markup)

[3] "soc" stands for "summary of changes".

5 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.

5.1 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.

5.2 Footnotes and margin notes

There is a problem of typesetting footnotes or margin notes in special environments, such as tables or tabbings. Avoid such markup when using these environments.

5.3 The *ulem* package

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

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

In that case there are only a few good solutions, the best way is to avoid using the *ulem* package by defining your own deletion markup. See

– Section 4.1.5
– Section 4.6.2

6 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
– Mittelbach, Frank
– Voss, Herbert
– Wölfel, Philipp
– Wolter, Steve

7 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`

8 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/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
tex/latex/changes/changes.sty
```

9 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⟩
```

9.1 Package information and options

Set needed LaTeX-format to LaTeX 2_ε, provide name, date, version. Type some information to the console.

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{changes}
4 [2019/07/21 v3.1.3 changes package]
5 \typeout{*** changes package 2019/07/21 v3.1.3 ***}
```

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

```
6 \RequirePackage{xkeyval}
```

Package *xifthen* provides improved `if` as well as a `while`-loop.

```
7 \RequirePackage{xifthen}
```

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

```
8 \RequirePackage{xstring}
```

9.1.1 Package options

Option `draft`, *default* is `true`.

```
9 \newboolean{Changes@optiondraft}
10 \setboolean{Changes@optiondraft}{true}
11 \DeclareOptionX{draft}{
12 \setboolean{Changes@optiondraft}{true}
13 \typeout{changes-option '\CurrentOption'}
14 }
```

Option `final`, sets `draft` to `false`.

```
15 \DeclareOptionX{final}{
16 \setboolean{Changes@optiondraft}{false}
17 \typeout{changes-option '\CurrentOption'}
18 }
```

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.

```
\newcommand{\Changes@optionaddedmarkup}{colored}
\newcommand{\Changes@optiondeletedmarkup}{sout}
\newcommand{\Changes@optionhighlightmarkup}{background}
\newcommand{\Changes@optioncommentmarkup}{todo}
```

Option markup, sets markup options accordingly.

```
\newcommand{\Changes@optionmarkup}{default}
\DeclareOptionX{markup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{default}\or
\equal{#1}{underlined}\or
\equal{#1}{bfit}\or
\equal{#1}{nocolor}
}
{\renewcommand{\Changes@optionmarkup}{#1}}
{\PackageWarning{changes}{markup '#1' unknown, using '\Changes@optionmarkup'}}
}
\ifthenelse{\equal{\Changes@optionmarkup}{default}}
{
% nothing to do
}
{}
\ifthenelse{\equal{\Changes@optionmarkup}{underlined}}
{
\renewcommand{\Changes@optionaddedmarkup}{uline}
\renewcommand{\Changes@optionhighlightmarkup}{uwave}
}
{}
\ifthenelse{\equal{\Changes@optionmarkup}{bfit}}
{
\renewcommand{\Changes@optionaddedmarkup}{bf}
\renewcommand{\Changes@optiondeletedmarkup}{it}
}
{}
\ifthenelse{\equal{\Changes@optionmarkup}{nocolor}}
{
\renewcommand{\Changes@optionaddedmarkup}{uline}
\renewcommand{\Changes@optionhighlightmarkup}{uwave}
}
{}
```

```
\typeout{changes-option 'markup=\Changes@optionmarkup'}
}
```

Option addedmarkup, stored or set to default value “colored”.

```
\DeclareOptionX{addedmarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{colored}\or
\equal{#1}{uline}\or
\equal{#1}{uuline}\or
\equal{#1}{uwave}\or
\equal{#1}{dashuline}\or
\equal{#1}{dotuline}\or
\equal{#1}{bf}\or
\equal{#1}{it}\or
\equal{#1}{sl}\or
\equal{#1}{em}
}{\renewcommand{\Changes@optionaddedmarkup}{#1}}
{\PackageWarning{changes}{addedmarkup '#1' unknown, using '\Changes@optionaddedmarkup'}}
}
\typeout{changes-option 'addedmarkup=\Changes@optionaddedmarkup'}
}
```

Option deletedmarkup, stored or set to default value “sout”.

```
\DeclareOptionX{deletedmarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{sout}\or
\equal{#1}{colored}\or
\equal{#1}{uline}\or
\equal{#1}{uuline}\or
\equal{#1}{uwave}\or
\equal{#1}{dashuline}\or
\equal{#1}{dotuline}\or
\equal{#1}{xout}\or
\equal{#1}{bf}\or
\equal{#1}{it}\or
\equal{#1}{sl}\or
\equal{#1}{em}
}{\renewcommand{\Changes@optiondeletedmarkup}{#1}}
{\PackageWarning{changes}{deletedmarkup '#1' unknown, using '\Changes@optiondeletedmarkup'}}
}
```

```
\typeout{changes-option 'deletedmarkup=\Changes@optiondeletedmarkup'}
}
```

Option highlightmarkup, stored or set to default value “background”.

```
\DeclareOptionX{highlightmarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{background}\or
\equal{#1}{uuline}\or
\equal{#1}{uwave}
}
{\renewcommand{\Changes@optionhighlightmarkup}{#1}}
{\PackageWarning{changes}{highlightmarkup '#1' unknown, using '\Changes@optionhighlightmarkup'}}
}
\typeout{changes-option 'highlightmarkup=\Changes@optionhighlightmarkup'}
}
```

Option commentmarkup, stored or set to default value “todo”.

```
\DeclareOptionX{commentmarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{todo}\or
\equal{#1}{margin}\or
\equal{#1}{footnote}\or
\equal{#1}{uwave}
}
{\renewcommand{\Changes@optioncommentmarkup}{#1}}
{\PackageWarning{changes}{commentmarkup '#1' unknown, using '\Changes@optioncommentmarkup'}}
}
\typeout{changes-option 'commentmarkup=\Changes@optioncommentmarkup'}
}
```

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

```
\newcommand{\Changes@optionauthormarkup}{superscript}
\DeclareOptionX{authormarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{superscript}\or
\equal{#1}{subscript}\or
\equal{#1}{brackets}\or
```

```
\equal{#1}{footnote}\or
\equal{#1}{none}
}
{\renewcommand{\Changes@optionauthormarkup}{#1}}
{\PackageWarning{changes}{authormarkup '#1' unknown, using '\Changes@optionauthormarkup'}}
}
\typeout{changes-option 'authormarkup=\Changes@optionauthormarkup'}
}
```

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

```
\newcommand{\Changes@optionauthormarkupposition}{right}
\DeclareOptionX{authormarkupposition}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{right}\or
\equal{#1}{left}
}
{\renewcommand{\Changes@optionauthormarkupposition}{#1}}
{\PackageWarning{changes}{authormarkupposition '#1' unknown, using '\Changes@optionauthormarkup
}
\typeout{changes-option 'authormarkupposition=\Changes@optionauthormarkupposition'}
}
```

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

```
\newcommand{\Changes@optionauthormarkuptext}{id}
\DeclareOptionX{authormarkuptext}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{id}\or
\equal{#1}{name}
}
{\renewcommand{\Changes@optionauthormarkuptext}{#1}}
{\PackageWarning{changes}{authormarkuptext '#1' unknown, using '\Changes@optionauthormarkuptext
}
\typeout{changes-option 'authormarkuptext=\Changes@optionauthormarkuptext'}
}
```

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

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

```
}
```

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

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

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

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

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

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

Unknown options generate a package warning.

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

9.1.2 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.

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

Set the default values of the options.

```
\presetkeys{Changes@definechangesauthor}{
name=\@empty,
color=blue
}{}
```

\added

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@added}{
id=\@empty,
remark=\@empty,
comment=\@empty,
}{}
```

\deleted

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@deleted}{
id=\@empty,
remark=\@empty,
comment=\@empty,
}{}
```

\replaced

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@replaced}{
id=\@empty,
remark=\@empty,
comment=\@empty,
}{}
```

\highlight

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@highlight}{
id=\@empty,
remark=\@empty,
comment=\@empty,
}{}
```

\comment

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

\listofchanges

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@loc}{
style=list,
title=\@empty,
show=all,
}{}
```

9.1.3 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.

```
\ExecuteOptionsX{
ulem={normalem,normalbf},
truncate={breakall,fit}
}
```

9.1.4 Option processing

Process the options.

```
\ProcessOptionsX*\relax
```

9.2 Packages

\isColored Check if text should be colored.

```
\newtest{\isColored}{%
\not\equal{\Changes@optionmarkup}{nocolor}%
}
```

Package *xcolor* provides colored text. Package *pdfcolmk* solves the problem of colored text and page breaks (has to be loaded after *xcolor*).

```
\ifthenelse{\isColored}
{
\RequirePackage{xcolor}
\RequirePackage{pdfcolmk}
}
{}
```

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

```
\ifthenelse{
\equal{\Changes@optionaddedmarkup}{uline}\or
\equal{\Changes@optionaddedmarkup}{uuline}\or
\equal{\Changes@optionaddedmarkup}{uwave}\or
\equal{\Changes@optionaddedmarkup}{dashuline}\or
\equal{\Changes@optionaddedmarkup}{dotuline}\or
\equal{\Changes@optiondeletedmarkup}{uline}\or
\equal{\Changes@optiondeletedmarkup}{uuline}\or
```

```
\equal{\Changes@optiondeletedmarkup}{uwave}\or
\equal{\Changes@optiondeletedmarkup}{dashuline}\or
\equal{\Changes@optiondeletedmarkup}{dotuline}\or
\equal{\Changes@optiondeletedmarkup}{sout}\or
\equal{\Changes@optiondeletedmarkup}{xout}\or
\equal{\Changes@optioncommentmarkup}{uwave}\or
\equal{\Changes@optionhighlightmarkup}{uuline}\or
\equal{\Changes@optionhighlightmarkup}{uwave}
}
{\RequirePackage{ulem}}
{}
```

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

```
\ifthenelse{
\equal{\Changes@optioncommentmarkup}{todo}
}
{\RequirePackage{todonotes}}
{}
```

9.3 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.

```
\newcommand*\listofchangesname{List of changes}
\newcommand*\summaryofchangesname{Changes}
\newcommand*\compactsummaryofchangesname{Changes (compact)}
\newcommand*\changesaddname{Added}
\newcommand*\changesdeletename{Deleted}
\newcommand*\changesreplacename{Replaced}
\newcommand*\changeshighlightname{Highlighted}
\newcommand*\changescommentname{Commented}
\newcommand*\changesauthorname{Author}
\newcommand*\changesanonymousname{anonymous}
\newcommand*\changesnochanges{No changes.}
\newcommand*\changesnoloc{List of changes is available after the next \LaTeX\ run.}
\newcommand*\changesnosoc{Summary of changes is available after the next \LaTeX\ run.}
```

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

```
\newboolean{Changes@langpackage}
\setboolean{Changes@langpackage}{false}
\@ifpackageloaded{babel}
{\setboolean{Changes@langpackage}{true}}
{}
\@ifpackageloaded{polyglossia}
```

```
{\setboolean{Changes@langpackage}{true}}
{}
\ifthenelse{\boolean{Changes@langpackage}}
{
\addto\captionsngerman{\def\listofchangesname{Liste der \"Anderungen}}
\addto\captionsngerman{\def\summaryofchangesname{\"Anderungen}}
\addto\captionsngerman{\def\compactsummaryofchangesname{\"Anderungen (kompakt)}}
\addto\captionsngerman{\def\changesaddname{Eingef\"ugt}}
\addto\captionsngerman{\def\changesdeletename{Gel\"oscht}}
\addto\captionsngerman{\def\changesreplacename{Ersetzt}}
\addto\captionsngerman{\def\changeshighlightname{Hervorgehoben}}
\addto\captionsngerman{\def\changescommentname{Kommentiert}}
\addto\captionsngerman{\def\changesauthorname{Autor}}
\addto\captionsngerman{\def\changesanonymousname{Anonym}}
\addto\captionsngerman{\def\changesnochanges{Keine \"Anderungen.}}
\addto\captionsngerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \LaTeX-Lauf
\addto\captionsngerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \LaTeX-Lauf verf\"ugb

\addto\captionsgerman{\def\listofchangesname{Liste der \"Anderungen}}
\addto\captionsgerman{\def\summaryofchangesname{\"Anderungen}}
\addto\captionsgerman{\def\compactsummaryofchangesname{\"Anderungen (kompakt)}}
\addto\captionsgerman{\def\changesaddname{Eingef\"ugt}}
\addto\captionsgerman{\def\changesdeletename{Gel\"oscht}}
\addto\captionsgerman{\def\changesreplacename{Ersetzt}}
\addto\captionsgerman{\def\changeshighlightname{Hervorgehoben}}
\addto\captionsgerman{\def\changescommentname{Kommentiert}}
\addto\captionsgerman{\def\changesauthorname{Autor}}
\addto\captionsgerman{\def\changesanonymousname{Anonym}}
\addto\captionsgerman{\def\changesnochanges{Keine \"Anderungen.}}
\addto\captionsgerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \LaTeX-Lauf
\addto\captionsgerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \LaTeX-Lauf verf\"ugba

\addto\captionsenglish{\def\listofchangesname{List of changes}}
\addto\captionsenglish{\def\summaryofchangesname{Changes}}
\addto\captionsenglish{\def\compactsummaryofchangesname{Changes (compact)}}
\addto\captionsenglish{\def\changesaddname{Added}}
\addto\captionsenglish{\def\changesdeletename{Deleted}}
\addto\captionsenglish{\def\changesreplacename{Replaced}}
\addto\captionsenglish{\def\changeshighlightname{Highlighted}}
\addto\captionsenglish{\def\changescommentname{Commented}}
\addto\captionsenglish{\def\changesauthorname{Author}}
\addto\captionsenglish{\def\changesanonymousname{anonymous}}
\addto\captionsenglish{\def\changesnochanges{No changes.}}
\addto\captionsenglish{\def\changesnoloc{List of changes is available after the next \LaTeX\ ru
\addto\captionsenglish{\def\changesnosoc{Summary of changes is available after the next \LaTeX\

\addto\captionsbritish{\def\listofchangesname{List of changes}}
\addto\captionsbritish{\def\summaryofchangesname{Changes}}
\addto\captionsbritish{\def\compactsummaryofchangesname{Changes (compact)}}
```

```
\addto\captionsbritish{\def\changesaddname{Added}}
\addto\captionsbritish{\def\changesdeletename{Deleted}}
\addto\captionsbritish{\def\changesreplacename{Replaced}}
\addto\captionsbritish{\def\changeshighlightname{Highlighted}}
\addto\captionsbritish{\def\changescommentname{Commented}}
\addto\captionsbritish{\def\changesauthorname{Author}}
\addto\captionsbritish{\def\changesanonymousname{anonymous}}
\addto\captionsbritish{\def\changesnochanges{No changes.}}
\addto\captionsbritish{\def\changesnoloc{List of changes is available after the next \LaTeX\ ru
\addto\captionsbritish{\def\changesnosoc{Summary of changes is available after the next \LaTeX\
\addto\captionsitalian{\def\listofchangesname{Lista delle modifiche}}
\addto\captionsitalian{\def\summaryofchangesname{Modifiche}}
\addto\captionsitalian{\def\compactsummaryofchangesname{Modifiche (coerente)}} % translation by
\addto\captionsitalian{\def\changesaddname{Aggiunte}}
\addto\captionsitalian{\def\changesdeletename{Cancellazioni}}
\addto\captionsitalian{\def\changesreplacename{Sostituzioni}}
\addto\captionsitalian{\def\changeshighlightname{Accentare}} % translation by me (EK), please p
\addto\captionsitalian{\def\changescommentname{Commenti}} % translation by me (EK), please prov
\addto\captionsitalian{\def\changesauthorname{Autore}}
\addto\captionsitalian{\def\changesanonymousname{anonimo}}
\addto\captionsitalian{\def\changesnochanges{Nessuna modifica.}} % translation by me (EK), plea
\addto\captionsitalian{\def\changesnoloc{La lista delle modifiche sar\‘a disponibile alla pross
\addto\captionsitalian{\def\changesnosoc{Le modifiche sar\‘a disponibile alla prossima esecuzio
}
{}
```

9.4 File extension

anges@extension Store file extension in variable, set default to soc (summary of changes).

```
\newcommand{\Changes@extension}{soc}
```

setsocextension Set a new file extension. Argument: new extension.

```
\newcommand{\setsocextension}[1]{
\renewcommand{\Changes@extension}{#1}
}
```

9.5 Authors

9.5.1 Author management

Author counter.

```
\newcounter{Changes@AuthorCount}
```

```
\setcounter{Changes@AuthorCount}{0}
\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: blue).

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

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

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

```
\setkeys{Changes@definechangesauthor}{#1}
```

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

```
\stepcounter{Changes@AuthorCount}
```

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

```
\@namedef{Changes@AuthorID\theChanges@AuthorCount}{#2}
```

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

```
\expandafter\let\csname Changes@AuthorName#2\endcsname=\Changes@definechangesauthor@name
\expandafter\let\csname Changes@AuthorColor#2\endcsname=\Changes@definechangesauthor@color
\newcounter{Changes@addedCount#2}
\newcounter{Changes@deletedCount#2}
\newcounter{Changes@replacedCount#2}
\newcounter{Changes@highlightCount#2}
\newcounter{Changes@commentCount#2}
}
```

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

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

9.5.2 Author markup

s@Markup@author Store markup for authors.

```
\newcommand{\Changes@Markup@author}[1]{%
\ifthenelse{\equal{\Changes@optionauthormarkup}{superscript}}{\textsuperscript{#1}}{}%
\ifthenelse{\equal{\Changes@optionauthormarkup}{subscript}}{\textsubscript{#1}}{}%
\ifthenelse{\equal{\Changes@optionauthormarkup}{brackets}}{(#1)}{}%
\ifthenelse{\equal{\Changes@optionauthormarkup}{footnote}}{\footnote{#1}}{}%
\ifthenelse{\equal{\Changes@optionauthormarkup}{none}}{}{}%
}
```

setauthormarkup Set markup for authors.

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

rmarkupposition Set position for author markup text.

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

uthormarkuptext Set author markup text to be displayed.

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

9.6 Change management commands

9.6.1 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.

es@Markup@added Store markup for added text.

```
\newcommand{\Changes@Markup@added}[1]{%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{colored}}{#1}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{uline}}{\uline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{uuline}}{\uuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{uwave}}{\uwave{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{dashuline}}{\dashuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{dotuline}}{\dotuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{bf}}{\textbf{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{it}}{\textit{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{sl}}{\textsl{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{em}}{\emph{#1}}{}%
}
```

\setaddedmarkup Set markup for added text.

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

@Markup@deleted Store markup for deleted text.

```
\newcommand{\Changes@Markup@deleted}[1]{%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{sout}}{\sout{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{colored}}{#1}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{uline}}{\uline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{uuline}}{\uuline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{uwave}}{\uwave{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{dashuline}}{\dashuline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{dotuline}}{\dotuline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{xout}}{\xout{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{bf}}{\textbf{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{it}}{\textit{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{sl}}{\textsl{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{em}}{\emph{#1}}{}%
}
```

etdeletedmarkup Set markup for deleted text.

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

arkup@highlight Store markup for highlighted text.

```
\newcommand{\Changes@Markup@highlight}[1]{%
\ifthenelse{\equal{\Changes@optionhighlightmarkup}{background}}%
{%
\ifthenelse{\isColored}%
{\colorbox{authorcolor!30}{#1}}%
{#1}%
}{}%
\ifthenelse{\equal{\Changes@optionhighlightmarkup}{uuline}}{\uuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionhighlightmarkup}{uwave}}{\uwave{#1}}{}%
}
```

highlightmarkup Set markup for highlighted text.

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

@Markup@comment Store markup for comments.

Parameters:

1. text
2. author's id

3. author's id/name output

```
\newcommand{\Changes@Markup@comment}[3]{%
```

This one is tricky, because the parameters depend on tests. If I use the tests inside the \todo command, they break because of the use of *ifthenelse*. Thus I am implementing a slightly dirty working version, having in mind, that the code should be revisited in future releases.

```
\ifthenelse{\equal{\Changes@optioncommentmarkup}{todo}}%
{%
\ifthenelse{\isColored}%
{%
\ifthenelse{\isAnonymous{#2}}%
{%
\todo[color=authorcolor!10, bordercolor=authorcolor, linecolor=authorcolor!70, nolist]{\textbf{
}}{%
\todo[color=authorcolor!10, bordercolor=authorcolor, linecolor=authorcolor!70, nolist]{\textbf{
}%
}{%
\ifthenelse{\isAnonymous{#2}}%
{%
\todo[color=black!0, bordercolor=black, linecolor=black!70, nolist]{\textbf{[\arabic{authorcomm
}}{%
\todo[color=black!0, bordercolor=black, linecolor=black!70, nolist]{\textbf{[#3~\arabic{authorc
}%
}%
}{}%
```

Something a little more easy.

```
\ifthenelse{\equal{\Changes@optioncommentmarkup}{margin}}%
{%
\marginpar{%
\ifthenelse{\isColored}%
{\leavevmode\color{authorcolor}}%
{}%
\ifthenelse{\isAnonymous{#2}}%
{\textbf{[\arabic{Changes@commentCount#2}]:} }%
{\textbf{[#3~\arabic{Changes@commentCount#2}]:} }%
#1%
}%
}{}%
\ifthenelse{\equal{\Changes@optioncommentmarkup}{footnote}}%
{%
\footnote{%
\ifthenelse{\isAnonymous{#2}}%
{\textbf{[\arabic{Changes@commentCount#2}]:} }%
{\textbf{[#3~\arabic{Changes@commentCount#2}]:} }%
```

```
#1%
}%
}{}%
\ifthenelse{\equal{\Changes@optioncommentmarkup}{uwave}}%
{%
{%
\ifthenelse{\isColored}%
{\color{authorcolor}}%
{}%
\allowbreak%
\uwave{%
\ifthenelse{\isAnonymous{#2}}%
{\textbf{[\arabic{Changes@commentCount#2}]:} }%
{\textbf{[#3~\arabic{Changes@commentCount#2}]:} }%
#1%
}%
}%
}{}%
}
```

etcommentmarkup Set markup for comments.

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

9.6.2 Change management command definition

\ifIsEmpty Checks if text in #1 is empty, executes #2 if empty, #3 otherwise. This is a shortcut for the \ifthenelse test, it basically eases the use of the test.

```
\DeclareRobustCommand{\ifIsEmpty}[3]{%
\ifthenelse{\equal{#1}{} \or \equal{#1}{\@empty}}%
{#2}%
{#3}%
}
```

\isAnonymous Check if author id is empty, therefore the author is anonymous. This is a new test that can be tested using \ifthenelse.

This test has the following arguments:

1. author's id

```
\newtest{\isAnonymous}[1]{%
\equal{#1}{\@empty}%
}
```

\isAuthorEmpty Check if author is anonymous or position does not equal needed position, therefore the author text is empty. This is a new test that can be tested using \ifthenelse.

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

This test has the following arguments:

1. author's id
2. position

```
\newtest{\isAuthorEmpty}[2]{%
\isAnonymous{#1} \or \not\equal{\Changes@optionauthormarkupposition}{#2}%
}
```

es@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. I have the feeling that the code is optimizable.

This command has the following arguments:

1. author's id

```
\newboolean{Changes@WrongID}
\newcommand{\Changes@check@author}[1]{%
\ifIsEmpty{#1}%
{}%
{%
\setboolean{Changes@WrongID}{true}%
\setcounter{Changes@Author}{0}%
\whiledo{\value{Changes@Author} < \value{Changes@AuthorCount}}{%
\stepcounter{Changes@Author}%
\ifthenelse{\equal{#1}{\@nameuse{Changes@AuthorID\theChanges@Author}}%
{\setboolean{Changes@WrongID}{false}}%
{}%
}%
\ifthenelse{\boolean{Changes@WrongID}}%
{\PackageError{changes}%
{Undefined changes author: #1}%
{You have to define the author #1 with e.g.: \definechangesauthor{#1}}}%
{}%
}%
}
```

s@output@author Output command for the author.

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.

```
\DeclareRobustCommand{\Changes@output@author}[2]{%
```

Output author text only if author's id is given and the position matches, otherwise output empty text.

```
\ifthenelse{\isAuthorEmpty{#1}{#2}}%
{}%
{%
\ifthenelse{\equal{\Changes@optionauthormarkuptext}{id}}%
{%
#1%
}%
{}%
\ifthenelse{\equal{\Changes@optionauthormarkuptext}{name}}%
{%
\ifIsEmpty{\@nameuse{Changes@AuthorName#1}}%
{%
#1%
}{%
\@nameuse{Changes@AuthorName#1}%
}%
}%
{}%
}%
}
```

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

This command has the following argument:

1. author's id

```
\newcommand{\Changes@set@color}[1]{%
\ifthenelse{\isColored}%
{\colorlet{authorcolor}{\@nameuse{Changes@AuthorColor#1}}}%
{}%
}
```

et@commentcount Sets the author's comment count.

This command has the following argument:

1. author's id

```
\newcounter{authorcommentcount}
\newcommand{\Changes@set@commentcount}[1]{%
\setcounter{authorcommentcount}{\value{Changes@commentCount#1}}%
}
```

\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

```
\newcommand{\Changes@output}[7]{%
```

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

```
\ifthenelse{\boolean{Changes@optiondraft}}%
{%
```

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

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

Start output.

```
{%
```

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

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

```
\ifthenelse{%
\equal{#1}{added}\or%
\equal{#1}{deleted}\or%
\equal{#1}{replaced}\or%
\equal{#1}{highlight}%
}%
{%
```

Author text for left positioning (only without comment).

```
\ifIsEmpty{#5}%
{%
\ifthenelse{\isAuthorEmpty{#2}{left}}%
{}%
{{%
\ifthenelse{\isColored}%
{\color{authorcolor}}%
{}%
```

```
\Changes@Markup@author{\Changes@output@author{#2}{left}}%
}}%
}{}%
```

Changed/highlighted text.

```
{%
\ifthenelse{\not\equal{#1}{highlight}}%
{%
\ifthenelse{\isColored}%
{\color{authorcolor}}%
{}%
}{}%
\ifthenelse{\equal{#1}{added}}{\Changes@Markup@added{#3}}{}%
\ifthenelse{\equal{#1}{deleted}}{\Changes@Markup@deleted{#4}}{}%
\ifthenelse{\equal{#1}{replaced}}{{\Changes@Markup@added{#3}}\allowbreak\Changes@Markup@deleted
\ifthenelse{\equal{#1}{highlight}}{\Changes@Markup@highlight{#3}}{}%
}%
```

Author text for right positioning (only without comment).

```
\ifIsEmpty{#5}%
{%
\ifthenelse{\isAuthorEmpty{#2}{right}}%
{}%
{{%
\ifthenelse{\isColored}%
{\color{authorcolor}}%
{}%
\Changes@Markup@author{\Changes@output@author{#2}{right}}%
}}%
}{}%
```

Update counters.

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

Comments.

```
\ifIsEmpty{#5}%
{}%
{%
\stepcounter{Changes@commentCount#2}%
\Changes@set@commentcount{#2}%
\Changes@Markup@comment%
{#5}%
{#2}%
{\Changes@output@author{#2}{left}\Changes@output@author{#2}{right}}%
}%
}%
```

Store line for list of changes.

```
\ifIsEmpty{#2}%
{\def\Changes@locid{}}%
{\def\Changes@locid{~(#2)}}%
\addtocontents{loc}{\protect\ChangesListline{#1}{#6\Changes@locid}{#7}{\thepage}}%
}%
```

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

```
{%
\ifIsEmpty{#3}%
{\@bsphack\@esphack}%
{#3}%
}%
}
```

\added The command formats text as new text.

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

```
\newcommand{\added}[2][\@empty]{%
```

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

```
\setkeys{Changes@added}{#1}%
```

Check for use of deprecated `remark` option.

```
\ifIsEmpty{\Changes@added@remark}%
{}%
{%
\ifIsEmpty{\Changes@added@comment}%
{%
\PackageWarning{changes}{You used the deprecated option 'remark' in your markup, please use 'co
\let\Changes@added@comment\Changes@added@remark%
}%
{%
\PackageWarning{changes}{You used both options 'comment' and the deprecated 'remark' in your ma
}%
}%
```

End of check for use of deprecated `remark` option.

```
\Changes@output%
{added}%
{\Changes@added@id}%
{#2}%
{}%
```

```
{\Changes@added@comment}%
{\changesaddname}%
{#2}%
}
```

\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, remark (deprecated)

```
\newcommand{\deleted}[2][\@empty]{%
```

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

```
\setkeys{Changes@deleted}{#1}%
```

Check for use of deprecated `remark` option.

```
\ifIsEmpty{\Changes@deleted@remark}%
{}%
{%
\ifIsEmpty{\Changes@deleted@comment}%
{%
\PackageWarning{changes}{You used the deprecated option 'remark' in your markup, please use 'co
\let\Changes@deleted@comment\Changes@deleted@remark%
}%
{%
\PackageWarning{changes}{You used both options 'comment' and the deprecated 'remark' in your ma
}%
}%
```

End of check for use of deprecated `remark` option.

```
\Changes@output%
{deleted}%
{\Changes@deleted@id}%
{}%
{#2}%
{\Changes@deleted@comment}%
{\changesdeletename}%
{#2}%
}
```

\replaced The command formats text as replaced text.

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

```
\newcommand{\replaced}[3][\@empty]{%
```

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

```
\setkeys{Changes@replaced}{#1}%
```

Check for use of deprecated remark option.

```
\ifIsEmpty{\Changes@replaced@remark}%
{}%
{%
\ifIsEmpty{\Changes@replaced@comment}%
{%
\PackageWarning{changes}{You used the deprecated option 'remark' in your markup, please use 'co
\let\Changes@replaced@comment\Changes@replaced@remark%
}%
{%
\PackageWarning{changes}{You used both options 'comment' and the deprecated 'remark' in your ma
}%
}%
```

End of check for use of deprecated remark option.

```
\Changes@output%
{replaced}%
{\Changes@replaced@id}%
{#2}%
{#3}%
{\Changes@replaced@comment}%
{\changesreplacename}%
{#2}%
}
```

\highlight The command formats text as highlighted text.

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

```
\newcommand{\highlight}[2][\@empty]{%
```

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

```
\setkeys{Changes@highlight}{#1}%
```

Check for use of deprecated remark option.

```
\ifIsEmpty{\Changes@highlight@remark}%
{}%
{%
\ifIsEmpty{\Changes@highlight@comment}%
{%
\PackageWarning{changes}{You used the deprecated option 'remark' in your markup, please use 'co
\let\Changes@highlight@comment\Changes@highlight@remark%
}%
{%
\PackageWarning{changes}{You used both options 'comment' and the deprecated 'remark' in your ma
}%
}%
```

End of check for use of deprecated remark option.

```
\Changes@output%
{highlight}%
{\Changes@highlight@id}%
{#2}%
{}%
{\Changes@highlight@comment}%
{\changeshighlightname}%
{#2}%
}
```

\comment The command formats text as comment.

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

```
\newcommand{\comment}[2][\@empty]{%
```

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

```
\setkeys{Changes@comment}{#1}%
\Changes@output%
{comment}%
{\Changes@comment@id}%
{}%
{}%
{#2}%
{\changescommentname}%
{#2}%
}
```

9.7 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.

```
\RequirePackage{truncate}
\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.

```
\def\changes@chopline#1;#2;#3;#4;#5;#6;#7;#8 \\{%
\def\Changes@InID{#1}%
\def\Changes@InColor{#2}%
\def\Changes@InName{#3}%
\def\Changes@InAdded{#4}%
\def\Changes@InDeleted{#5}%
\def\Changes@InReplaced{#6}%
\def\Changes@InHighlight{#7}%
\def\Changes@InComment{#8}%
}
```

ChangesListline Output of a list line.

This command has the following arguments:

1. change type (added, ...)
2. text
3. page

```
\newcommand{\ChangesListline}[4]{%
\IfSubStr{\Changes@loc@show}{#1}{%
\@ifundefined{@dotsep}%
{\def\@dotsep{4.5}}{}%
\@dottedtocline{1}{0px}{2em}{#2: \truncate{\Changes@truncate@width}{#3}}{#4}%
}{}%
}
```

@truncate@width Length for the width of the truncation.

Default: two third of the text width

```
\newlength{\Changes@truncate@width}
\setlength{\Changes@truncate@width}{.6\textwidth}
```

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

```
\newcommand{\settruncatewidth}[1]{
\setlength{\Changes@truncate@width}{#1}
}
```

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

Default: one third of the text width

```
\newlength{\Changes@summary@width}
\setlength{\Changes@summary@width}{.3\textwidth}
```

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

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

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

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

ges@summaryline Auxiliary command for output of a summary line.

This command has the following arguments:

1. change type (added, ...)
2. number of items
3. name of items
4. line delimiter

```
\newcommand{\Changes@summaryline}[4]{%
\IfSubStr{\Changes@loc@show}{#1}{%
\ifthenelse{\not\equal{\Changes@loc@style}{compactsummary} \or #2 > 0}%
{%
\parbox{\Changes@summary@width}{#3~\let\cleaders\leaders\dotfill~#2}#4%
}{}%
}{}%
}
```

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

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

For the list, the values are read from the auxiliary file.

For the summary, the values are read from the loc-file, if it exists. If no loc-file exists, an according message is generated.

Some definitions that have to reside outside the command in order to use the command multiple times.

```
\newboolean{Changes@MoreLines}
\newboolean{Changes@ShowOK}
```

The definition of `\listofchanges`.

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

All work is done only in draft mode.

```
\ifthenelse{\boolean{Changes@optiondraft}}%
{%
```

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

```
\ifIsEmpty{\Changes@loc@style}%
{\def\Changes@loc@style{list}}%
{%
\ifthenelse{%
\equal{\Changes@loc@style}{list}\or%
\equal{\Changes@loc@style}{summary}\or%
\equal{\Changes@loc@style}{compactsummary}%
}%
{}%
{%
\PackageWarning{changes}{Wrong style for list of changes: '\Changes@loc@style', using 'list' in
\def\Changes@loc@style{list}%
}%
}%
```

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

```
\ifIsEmpty{\Changes@loc@show}%
{\def\Changes@loc@show{all}}%
{%
```

This check is complicated, because the \isin test of *xifthen* does not work with macros. On the other hand I could not define a new text using the \IfSubStr macro of *xstring*,

```
\setboolean{Changes@ShowOK}{false}%
\ifthenelse{\equal{\Changes@loc@show}{all}}{\setboolean{Changes@ShowOK}{true}}{}%
\IfSubStr{\Changes@loc@show}{added}{\setboolean{Changes@ShowOK}{true}}{}%
\IfSubStr{\Changes@loc@show}{deleted}{\setboolean{Changes@ShowOK}{true}}{}%
\IfSubStr{\Changes@loc@show}{replaced}{\setboolean{Changes@ShowOK}{true}}{}%
\IfSubStr{\Changes@loc@show}{highlight}{\setboolean{Changes@ShowOK}{true}}{}%
\IfSubStr{\Changes@loc@show}{comment}{\setboolean{Changes@ShowOK}{true}}{}%
\ifthenelse{\boolean{Changes@ShowOK}}%
{}%
{%
\PackageWarning{changes}{Wrong show-value for list of changes: '\Changes@loc@show', using 'all'
\def\Changes@loc@show{all}%
}%
}%
\ifthenelse{\equal{\Changes@loc@show}{all}}%
{%
\def\Changes@loc@show{added|deleted|replaced|highlight|comment}%
}{}%
```

Print heading.

```
\def\Changes@heading{\Changes@loc@title}
\ifIsEmpty{\Changes@loc@title}%
{%
\ifthenelse{\equal{\Changes@loc@style}{list}}%
{\def\Changes@heading{\listofchangesname}}{}%
\ifthenelse{\equal{\Changes@loc@style}{summary}}%
{\def\Changes@heading{\summaryofchangesname}}{}%
\ifthenelse{\equal{\Changes@loc@style}{compactsummary}}%
{\def\Changes@heading{\compactsummaryofchangesname}}{}%
}{}%
\section*{\Changes@heading}
```

Print list.

```
\ifthenelse{\equal{\Changes@loc@style}{list}}%
{%
\IfFileExists{\jobname.loc}%
{%
\setboolean{Changes@MoreLines}{true}%
\newread\Changes@InFile%
\openin\Changes@InFile = \jobname.loc%
\whiledo{\boolean{Changes@MoreLines}}{%
\read\Changes@InFile to \Changes@Line%
\ifeof\Changes@InFile%
\setboolean{Changes@MoreLines}{false}%
\else%
```

```
\Changes@Line%
\fi%
}%
\closein\Changes@InFile%
}{%
\emph{\changesnoloc}%
\PackageWarning{changes}{LaTeX rerun needed for list of changes}%
}%
}{}%
```

Print summary or compact summary.

```
\ifthenelse{\equal{\Changes@loc@style}{summary} \or \equal{\Changes@loc@style}{compactsummary}}
{%
\IfFileExists{\jobname.\Changes@extension}%
{%
\setboolean{Changes@MoreLines}{true}%
\newread\Changes@InFile%
\openin\Changes@InFile = \jobname.\Changes@extension%
\whiledo{\boolean{Changes@MoreLines}}{%
\read\Changes@InFile to \Changes@Line%
\ifeof\Changes@InFile%
\setboolean{Changes@MoreLines}{false}%
\else%
\expandafter\changes@chopline\Changes@Line\\%
\textbf{%
\ifthenelse{\isColored}%
{\color{\Changes@InColor}}%
{}%
\ifthenelse{\equal{\Changes@InID}{\@empty}}%
{\changesauthorname: \changesanonymousname}%
{%
\changesauthorname: \Changes@InID%
\ifthenelse{\equal{\Changes@InName}{\@empty}}%
{}%
{ (\Changes@InName)}%
}%
}\\%
\ifthenelse{%
\Changes@InAdded > 0 \or%
\Changes@InDeleted > 0 \or%
\Changes@InReplaced > 0 \or%
\Changes@InHighlight > 0 \or%
\Changes@InComment > 0%
}%
{%
\Changes@summaryline{added}{\Changes@InAdded}{\changesaddname}{\\}%
\Changes@summaryline{deleted}{\Changes@InDeleted}{\changesdeletename}{\\}%
\Changes@summaryline{replaced}{\Changes@InReplaced}{\changesreplacename}{\\}%
\Changes@summaryline{highlight}{\Changes@InHighlight}{\changeshighlightname}{\\}%
```

```
\Changes@summaryline{comment}{\Changes@InComment}{\changescommentname}{\\[1ex]}%
}%
{%
\parbox{\Changes@summary@width}{\changesnochanges}\\[1ex]%
}%
\fi%
}%
\closein\Changes@InFile%
}{%
\emph{\changesnosoc}%
\PackageWarning{changes}{LaTeX rerun needed for summary of changes}%
}%
}{}%
}
```

In final mode print nothing.

```
}{}%
}
```

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.

```
\AtEndDocument{%
```

Open output file.

```
\newwrite\Changes@OutFile
\immediate\openout\Changes@OutFile = \jobname.\Changes@extension
```

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

```
\let\Changes@protect\protect
\let\protect\@unexpandable@protect
```

Output data for list of changes.

```
\setcounter{Changes@Author}{0}
\whiledo{\value{Changes@Author} < \value{Changes@AuthorCount}}{%
\stepcounter{Changes@Author}%
\def\Changes@ID{\@nameuse{Changes@AuthorID\theChanges@Author}}%
\immediate\write\Changes@OutFile{\Changes@ID;%
\@nameuse{Changes@AuthorColor\Changes@ID};%
\@nameuse{Changes@AuthorName\Changes@ID};%
\the\value{Changes@addedCount\Changes@ID};%
\the\value{Changes@deletedCount\Changes@ID};%
```

```
\the\value{Changes@replacedCount\Changes@ID};%
\the\value{Changes@highlightCount\Changes@ID};%
\the\value{Changes@commentCount\Changes@ID}}%
}%
```

Close output file.

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

Restore original definition of \protect.

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

Write content of listofchanges to file.

```
\if@filesw
\@ifundefined{tf@loc}{%
\expandafter\newwrite\csname tf@loc\endcsname
\immediate\openout \csname tf@loc\endcsname \jobname.loc\relax
}{}%
\fi
}
```

⟨/changes⟩