The *changes*-package

Manual change markup — version 4.0.0

January 28, 2021

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 commandnameprefix 10

4.1.4 markup 11

4.1.5 addedmarkup 12

4.1.6 deletedmarkup 12

4.1.7 highlightmarkup 13

4.1.8 commentmarkup 13

4.1.9 authormarkup 14

4.1.10 authormarkupposition 15

4.1.11 authormarkuptext 15

4.1.12 defaultcolor 15

4.1.13 todonotes 16

4.1.14 truncate 16

4.1.15 ulem 16

4.1.16 xcolor 17

4.2 Change management 17

4.2.1 `\added` 17

4.2.2 `\deleted` 18

4.2.3 `\replaced` 18

4.3 Highlighting and Comments 19

4.3.1 `\highlight` 19

4.3.2 `\comment` 19

4.4 Overview of changes 20
4.4.1 \listofchanges 20
4.5 Author management 21
4.5.1 \definechangesauthor 21
4.6 Adaptation of the output 21
4.6.1 \setaddedmarkup 22
4.6.2 \setdeletedmarkup 22
4.6.3 \sethighlightmarkup 23
4.6.4 \setcommentmarkup 23
4.6.5 \setauthormarkup 24
4.6.6 \setauthormarkupposition 24
4.6.7 \setauthormarkuptext 25
4.6.8 \settruncatewidth 25
4.6.9 \setsummarywidth 25
4.6.10 \setsummarytowidth 26
4.6.11 \setlocextension 26
4.6.12 \setsocextension 26
4.7 Used packages 27

5 Remove markup from file 28

6 Known problems and solutions 29
6.1 Special content 29
6.2 Footnotes and margin notes 29
6.3 The *ulem* package 29
6.4 Command already defined 30

7 Authors 31

8 Versions 32

9 Distribution, Copyright, License **33**

10 The documented sourcecode **34**

10.1 Package information and options 34

10.1.1 Package options 35

10.1.2 Command options 40

10.1.3 Package options 42

10.1.4 Option processing 42

10.2 Utility tests 42

10.3 Packages 43

10.4 Language dependent texts 44

10.5 File extensions 46

10.6 Authors 47

10.6.1 Author management 47

10.6.2 Author markup 48

10.7 Change management commands 48

10.7.1 Text markup definition 48

10.7.2 Change management command definition 51

10.8 List of changes 58

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
– 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.

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 10
4.1.2 final 10
4.1.3 commandnameprefix 10
4.1.4 markup 11
4.1.5 addedmarkup 12
4.1.6 deletedmarkup 12
4.1.7 highlightmarkup 13
4.1.8 commentmarkup 13
4.1.9 authormarkup 14
4.1.10 authormarkupposition 15
4.1.11 authormarkuptext 15
4.1.12 defaultcolor 15
4.1.13 todonotes 16
4.1.14 truncate 16
4.1.15 ulem 16
4.1.16 xcolor 17

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 – 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}
```

4.1.7 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 – example
`uwave`	wavy underlined text – example

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:

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}
```

4.1.9 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$^{\text{author}}$ (default)
`left`	left of the text – $^{\text{author}}$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$^{\text{id}}$ (default)
`name`	author's name – text$^{\text{authorname}}$

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 17
4.2.2 \deleted 18
4.2.3 \replaced 18

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

[EK 5] better

[3] improved

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

4.3 Highlighting and Comments

4.3.1 \highlight 19
4.3.2 \comment 19

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 \setaddedmarkup 22
4.6.2 \setdeletedmarkup 22
4.6.3 \sethighlightmarkup 23
4.6.4 \setcommentmarkup 23
4.6.5 \setauthormarkup 24

4.6.6 \setauthormarkupposition 24
4.6.7 \setauthormarkuptext 25
4.6.8 \settruncatewidth 25
4.6.9 \setsummarywidth 25
4.6.10 \setsummarytowidth 26
4.6.11 \setlocextension 26
4.6.12 \setsocextension 26

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^{id} (default)
name	author's name – 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 \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.

4.6.12 \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

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

```
\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.

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 tabbings. Avoid such markup when using these environments.

6.3 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 4.1.6
– Section 4.6.2

6.4 Command already defined

Some package define the markup commands as well, especially `\comment` and `\highlight` are not exclusively used by *changes*.

If needed, *changes* can prefix it's markup commands in order to avoid name clashes. This is set with the option `commandnameprefix`, see Section 4.1.3 for the documentation.

In order for this to work, the *changes* package has to be used after the other package defining the command.

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
– 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/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
```

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.

```
⟨*changes⟩
```

10.1 Package information and options

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

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{changes}
[2021/01/28 v4.0.0 changes package]
\typeout{*** changes package 2021/01/28 v4.0.0 ***}
```

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

```
\RequirePackage{xkeyval}
```

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

```
\RequirePackage{etoolbox}
```

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

```
\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.

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

A temporary list and flag.

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

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

```
\newrobustcmd{\IfIsInList}[4]{%
\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`.

```
\renewcommand*{\do}[1]{%
\listadd{\Changes@tmplist}{##1}%
}%
\expandafter\dopsvlist\expandafter{#2}%
```

End fo the workaround for `\forpsvlist`.

```
\setbool{Changes@inlist}{false}%
\renewcommand*{\do}[1]{%
\ifinlist{##1}{\Changes@tmplist}%
{\setbool{Changes@inlist}{true}}%
{}%
}%
\expandafter\dopsvlist\expandafter{#1}%
\ifbool{Changes@inlist}%
{#3}%
{#4}%
}
```

10.1.1 Package options

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

```
\newbool{Changes@optiondraft}
\setbool{Changes@optiondraft}{true}
\DeclareOptionX{draft}{
\setbool{Changes@optiondraft}{true}
\typeout{changes-option '\CurrentOption'}
}
```

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

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

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

```
\newcommand{\Changes@optioncommandnameprefix}{none}
\DeclareOptionX{commandnameprefix}{
\ifblank{#1}
{}
```

```
{
\IfIsInList{#1}{none|always|ifneeded}
{\renewcommand{\Changes@optioncommandnameprefix}{#1}}
{\PackageWarning{changes}{commandnameprefix '#1' unknown, using '\Changes@optioncommandnamep
}
\typeout{changes-option 'commandnameprefix=\Changes@optioncommandnameprefix'}
}
```

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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{bfit|default|nocolor|underlined}
{\renewcommand{\Changes@optionmarkup}{#1}}
{\PackageWarning{changes}{markup '#1' unknown, using '\Changes@optionmarkup'}}
}
\IfStrEq{\Changes@optionmarkup}{default}
{
% nothing to do, included for symmetry and potential later use
}
{}
\IfStrEq{\Changes@optionmarkup}{underlined}
{
\renewcommand{\Changes@optionaddedmarkup}{uline}
\renewcommand{\Changes@optionhighlightmarkup}{uwave}
}
{}
\IfStrEq{\Changes@optionmarkup}{bfit}
{
\renewcommand{\Changes@optionaddedmarkup}{bf}
\renewcommand{\Changes@optiondeletedmarkup}{it}
}
{}
\IfStrEq{\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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{bf|colored|dashuline|dotuline|em|it|sl|uline|uuline|uwave}
{\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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{bf|colored|dashuline|dotuline|em|it|sl|sout|uline|uuline|uwave|xout}
{\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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{background|uuline|uwave}
{\renewcommand{\Changes@optionhighlightmarkup}{#1}}
{\PackageWarning{changes}{highlightmarkup '#1' unknown, using '\Changes@optionhighlightmarku
}
\typeout{changes-option 'highlightmarkup=\Changes@optionhighlightmarkup'}
}
```

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

```
\DeclareOptionX{commentmarkup}{
\ifblank{#1}
{}
```

```
{
\IfIsInList{#1}{footnote|margin|todo|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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{brackets|footnote|none|subscript|superscript}
{\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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{left|right}
{\renewcommand{\Changes@optionauthormarkupposition}{#1}}
{\PackageWarning{changes}{authormarkupposition '#1' unknown, using '\Changes@optionauthormar
}
\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}{
\ifblank{#1}
{}
{
\IfIsInList{#1}{id|name}
{\renewcommand{\Changes@optionauthormarkuptext}{#1}}
{\PackageWarning{changes}{authormarkuptext '#1' unknown, using '\Changes@optionauthormarkupt
}
```

```
\typeout{changes-option 'authormarkuptext=\Changes@optionauthormarkuptext'}
}
```

Store default author color..

```
\newcommand{\Changes@optiondefaultcolor}{blue}
\DeclareOptionX{defaultcolor}{
\ifblank{#1}
{}
{
\renewcommand{\Changes@optiondefaultcolor}{#1}
}
\typeout{changes-option 'defaultcolor=\Changes@optiondefaultcolor'}
}
```

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'}
}
```

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.

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

Set the default values of the options.

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

`\added`

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@added}{
id=\@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>{comment}{\def\Changes@deleted@comment{#1}}
```

Set the default values of the options.

```
\presetkeys{Changes@deleted}{
id=\@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>{comment}{\def\Changes@replaced@comment{#1}}
```

Set the default values of the options.

```
\presetkeys{Changes@replaced}{
id=\@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>{comment}{\def\Changes@highlight@comment{#1}}
```

Set the default values of the options.

```
\presetkeys{Changes@highlight}{
id=\@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,
}{}
```

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.

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

10.1.4 Option processing

Process the options.

```
\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.

```
\newrobustcmd{\IfIsColored}[2]{%
\IfStrEq{\Changes@optionmarkup}{nocolor}%
{#2}%
{#1}%
}
```

`\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.

```
\newrobustcmd{\IfIsEmpty}[3]{%
\ifboolexpr{%
```

```
test {\ifblank{#1}} or%
test {\IfStrEq{#1}{}} or%
test {\IfStrEq{#1}{\@empty}}%
}%
{#2}%
{#3}%
}
```

\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.

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

EmptyAtPosition 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. 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.

```
\newrobustcmd{\IfIsAuthorEmptyAtPosition}[4]{%
\ifboolexpr{%
test {\IfIsAnonymous{#1}} or%
not test {\IfStrEq{\Changes@optionauthormarkupposition}{#2}}%
}%
{#3}%
{#4}%
}
```

10.3 Packages

Package *xcolor* provides colored text.

```
\IfIsColored
{\RequirePackage{xcolor}}
{}
```

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

```
\ifboolexpr{
test {\IfIsInList{\Changes@optionaddedmarkup}{dashuline|dotuline|sout|uline|uuline|uwave|xou
```

```
\test {\IfIsInList{\Changes@optiondeletedmarkup}{dashuline|dotuline|sout|uline|uuline|uwave|x
\test {\IfIsInList{\Changes@optionhighlightmarkup}{dashuline|dotuline|sout|uline|uuline|uwave
}
{\RequirePackage{ulem}}
{}
```

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

```
\IfStrEq{\Changes@optioncommentmarkup}{todo}
{\RequirePackage{todonotes}}
{}
```

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.

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

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

```
\ifboolexpr{
test {\@ifpackageloaded{babel}} or
test {\@ifpackageloaded{polyglossia}}
}
{
\addto\captionsngerman{\def\listofchangesname{Liste der \"Anderungen}}
\addto\captionsngerman{\def\summaryofchangesname{\"Anderungen}}
\addto\captionsngerman{\def\compactsummaryofchangesname{\"Anderungen (kompakt)}}
\addto\captionsngerman{\def\changesaddedname{Eingef\"ugt}}
\addto\captionsngerman{\def\changesdeletedname{Gel\"oscht}}
\addto\captionsngerman{\def\changesreplacedname{Ersetzt}}
\addto\captionsngerman{\def\changeshighlightname{Hervorgehoben}}
\addto\captionsngerman{\def\changescommentname{Kommentiert}}
\addto\captionsngerman{\def\changesauthorname{Autor:in}}
```

```
\addto\captionsngerman{\def\changesanonymousname{Anonym}}
\addto\captionsngerman{\def\changesnochanges{Keine \"Anderungen.}}
\addto\captionsngerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \La-
TeX-Lauf verf\"ugbar.}}
\addto\captionsngerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \La-
TeX-Lauf verf\"ugbar.}}

\addto\captionsgerman{\def\listofchangesname{Liste der \"Anderungen}}
\addto\captionsgerman{\def\summaryofchangesname{\"Anderungen}}
\addto\captionsgerman{\def\compactsummaryofchangesname{\"Anderungen (kompakt)}}
\addto\captionsgerman{\def\changesaddedname{Eingef\"ugt}}
\addto\captionsgerman{\def\changesdeletedname{Gel\"oscht}}
\addto\captionsgerman{\def\changesreplacedname{Ersetzt}}
\addto\captionsgerman{\def\changeshighlightname{Hervorgehoben}}
\addto\captionsgerman{\def\changescommentname{Kommentiert}}
\addto\captionsgerman{\def\changesauthorname{Autor:in}}
\addto\captionsgerman{\def\changesanonymousname{Anonym}}
\addto\captionsgerman{\def\changesnochanges{Keine \"Anderungen.}}
\addto\captionsgerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \La-
TeX-Lauf verf\"ugbar.}}
\addto\captionsgerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \La-
TeX-Lauf verf\"ugbar.}}

\addto\captionsenglish{\def\listofchangesname{List of changes}}
\addto\captionsenglish{\def\summaryofchangesname{Changes}}
\addto\captionsenglish{\def\compactsummaryofchangesname{Changes (compact)}}
\addto\captionsenglish{\def\changesaddedname{Added}}
\addto\captionsenglish{\def\changesdeletedname{Deleted}}
\addto\captionsenglish{\def\changesreplacedname{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 \La-
TeX\ run.}}
\addto\captionsenglish{\def\changesnosoc{Summary of changes is available af-
ter the next \LaTeX\ run.}}

\addto\captionsbritish{\def\listofchangesname{List of changes}}
\addto\captionsbritish{\def\summaryofchangesname{Changes}}
\addto\captionsbritish{\def\compactsummaryofchangesname{Changes (compact)}}
\addto\captionsbritish{\def\changesaddedname{Added}}
\addto\captionsbritish{\def\changesdeletedname{Deleted}}
\addto\captionsbritish{\def\changesreplacedname{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\ run.}}
\addto\captionsbritish{\def\changesnosoc{Summary of changes is available after the next \LaTeX\ run.}}

\addto\captionsitalian{\def\listofchangesname{Lista delle modifiche}}
\addto\captionsitalian{\def\summaryofchangesname{Modifiche}}
\addto\captionsitalian{\def\compactsummaryofchangesname{Modifiche (coerente)}} % translation by me (EK), please provide correct translation
\addto\captionsitalian{\def\changesaddedname{Aggiunte}}
\addto\captionsitalian{\def\changesdeletedname{Cancellazioni}}
\addto\captionsitalian{\def\changesreplacedname{Sostituzioni}}
\addto\captionsitalian{\def\changeshighlightname{Accentare}} % translation by me (EK), please provide correct translation
\addto\captionsitalian{\def\changescommentname{Commenti}} % translation by me (EK), please provide correct translation
\addto\captionsitalian{\def\changesauthorname{Autore}}
\addto\captionsitalian{\def\changesanonymousname{anonimo}}
\addto\captionsitalian{\def\changesnochanges{Nessuna modifica.}} % translation by me (EK), provide correct translation
\addto\captionsitalian{\def\changesnoloc{La lista delle modifiche sar\'a disponibile alla prossima esecuzione di \LaTeX.}}
\addto\captionsitalian{\def\changesnosoc{Le modifiche sar\'a disponibile alla prossima esecuzione di \LaTeX.}}
}
{}
```

10.5 File extensions

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

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

setlocextension Sets a new file extension for list of changes.

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

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

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

setsocextension Sets a new file extension for summary of changes.

```
\newcommand{\setsocextension}[1]{
```

```
\renewcommand{\Changes@socextension}{#1}
}
```

10.6 Authors

10.6.1 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: `\Changes@optiondefaultcolor` (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}
```

10.6.2 Author markup

s@Markup@author Store markup for authors.

```
\newcommand{\Changes@Markup@author}[1]{%
\IfStrEq{\Changes@optionauthormarkup}{superscript}%
{\textsuperscript{#1}}%
{}%
\IfStrEq{\Changes@optionauthormarkup}{subscript}%
{\textsubscript{#1}}%
{}%
\IfStrEq{\Changes@optionauthormarkup}{brackets}%
{(#1)}%
{}%
\IfStrEq{\Changes@optionauthormarkup}{footnote}%
{\footnote{#1}}%
{}%
\IfStrEq{\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}
}
```

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.

```
\newcommand{\Changes@Markup}[2]{%
\IfStrEq{#1}{background}%
{%
\IfIsColored%
{\colorbox{authorcolor!30}{#2}}%
{#2}%
}%
{}%
\IfStrEq{#1}{bf}%
{\textbf{#2}}%
{}%
\IfStrEq{#1}{colored}%
{#2}%
{}%
\IfStrEq{#1}{dashuline}%
{\dashuline{#2}}%
{}%
\IfStrEq{#1}{dotuline}%
{\dotuline{#2}}%
{}%
\IfStrEq{#1}{em}%
{\emph{#2}}%
{}%
\IfStrEq{#1}{it}%
{\textit{#2}}%
{}%
\IfStrEq{#1}{sl}%
{\textsl{#2}}%
{}%
\IfStrEq{#1}{sout}%
{\sout{#2}}%
{}%
\IfStrEq{#1}{uline}%
{\uline{#2}}%
{}%
\IfStrEq{#1}{uuline}%
{\uuline{#2}}%
{}%
\IfStrEq{#1}{uwave}%
{\uwave{#2}}%
{}%
\IfStrEq{#1}{xout}%
{\xout{#2}}%
{}%
}
```

es@Markup@added Store markup for added text.

```
\newcommand{\Changes@Markup@added}[1]{%
\Changes@Markup{\Changes@optionaddedmarkup}{#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]{%
\Changes@Markup{\Changes@optiondeletedmarkup}{#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]{%
\Changes@Markup{\Changes@optionhighlightmarkup}{#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]{%
\IfStrEq{\Changes@optioncommentmarkup}{todo}%
{%
\IfIsColored%
{\colorlet{Changes@todocolor}{authorcolor}}%
{\colorlet{Changes@todocolor}{black}}%
\todo[color=Changes@todocolor!10, bordercolor=Changes@todocolor, linecolor=Changes@todocolor
   list]{\textbf{[\IfIsAnonymous{#2}{}{#3~}\arabic{authorcommentcount}]} #1}%
}{}%
\IfStrEq{\Changes@optioncommentmarkup}{margin}%
{%
\marginpar{%
\IfIsColored%
{\leavevmode\color{authorcolor}}%
{}%
\textbf{[\IfIsAnonymous{#2}{}{#3~}\arabic{Changes@commentCount#2}]:} #1%
}%
}{}%
\IfStrEq{\Changes@optioncommentmarkup}{footnote}%
{%
\footnote{%
\textbf{[\IfIsAnonymous{#2}{}{#3~}\arabic{Changes@commentCount#2}]:} #1%
}%
}{}%
\IfStrEq{\Changes@optioncommentmarkup}{uwave}%
{%
{%
\IfIsColored%
{\color{authorcolor}}%
{}%
\allowbreak%
\uwave{%
\textbf{[\IfIsAnonymous{#2}{}{#3~}\arabic{Changes@commentCount#2}]:} #1%
}%
}%
}{}%
}
```

`etcommentmarkup` Set markup for comments.

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

10.7.2 Change management command definition

`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.

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

This command has the following arguments:

1. author's id

```
\newbool{Changes@WrongID}
\newcommand{\Changes@check@author}[1]{%
\IfIsEmpty{#1}%
{}%
{%
\setbool{Changes@WrongID}{true}%
\setcounter{Changes@Author}{0}%
\loop%
\stepcounter{Changes@Author}%
\IfStrEq{#1}{\csuse{Changes@AuthorID\theChanges@Author}}%
{\setbool{Changes@WrongID}{false}}%
{}%
\ifnum\theChanges@Author<\theChanges@AuthorCount%
\repeat%
\ifbool{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.

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

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

```
\IfIsAuthorEmptyAtPosition{#1}{#2}%
{}%
{%
\IfStrEq{\Changes@optionauthormarkuptext}{id}%
{#1}%
```

```
{}%
\IfStrEq{\Changes@optionauthormarkuptext}{name}%
{%
\IfIsEmpty{\csuse{Changes@AuthorName#1}}%
{#1}%
{\csuse{Changes@AuthorName#1}}%
}%
{}%
}%
}
```

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

This command has the following argument:

1. author's id

```
\newrobustcmd{\Changes@set@color}[1]{%
\IfIsColored%
{\colorlet{authorcolor}{\csuse{Changes@AuthorColor#1}}}%
{}%
}
```

et@commentcount Sets the author's comment count.

This command has the following argument:

1. author's id

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

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

```
\def\Changes@commandprefix{ch}
\newrobustcmd{\Changes@set@commandname}[1]{
\def\Changes@commandname{#1}
\IfStrEq{\Changes@optioncommandnameprefix}{always}
{
\def\Changes@commandname{\Changes@commandprefix#1}
\typeout{changes-package option commandnameprefix=always: using prefix on #1, new com-
   mand name is \Changes@commandname.}
}{
\ifcsdef{#1}
{
\IfStrEq{\Changes@optioncommandnameprefix}{none}
{
```

```
\PackageError{changes}
{Command #1 is already defined.}
{Try package option commandnameprefix with "always" or "ifneeded", e.g. usepackage[commandnameprefix=always]{changes}}
}{}
\IfStrEq{\Changes@optioncommandnameprefix}{ifneeded}
{
\def\Changes@commandname{\Changes@commandprefix#1}
\PackageWarning{changes}{Command #1 is already defined, using \Changes@commandname}
}{}
}
{}
}
}
```

\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

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

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

```
\ifbool{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.

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

Author text for left positioning (only without comment).

```
\IfIsEmpty{#5}%
{%
\IfIsAuthorEmptyAtPosition{#2}{left}%
{}%
{{%
\IfIsColored%
{\color{authorcolor}}%
{}%
\Changes@Markup@author{\Changes@output@author{#2}{left}}%
}}%
}{}%
```

Changed/highlighted text.

```
{%
\IfStrEq{#1}{highlight}%
{}{%
\IfIsColored%
{\color{authorcolor}}%
{}%
}%
\IfStrEq{#1}{added}{\Changes@Markup@added{#3}}{}%
\IfStrEq{#1}{deleted}{\Changes@Markup@deleted{#4}}{}%
\IfStrEq{#1}{replaced}{{\Changes@Markup@added{#3}}\allowbreak\Changes@Markup@deleted{#4}}{}%
\IfStrEq{#1}{highlight}{\Changes@Markup@highlight{#3}}{}%
}%
```

Author text for right positioning (only without comment).

```
\IfIsEmpty{#5}%
{%
\IfIsAuthorEmptyAtPosition{#2}{right}%
{}%
{{%
\IfIsColored%
{\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{\Changes@locextension}{\protect\ChangesListline{#1}{#6\Changes@locid}{#7}{\th
}%
```

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

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

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

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

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

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

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

`\replaced` The command formats text as replaced text.

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

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

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

```
\setkeys{Changes@replaced}{#1}%
\Changes@output%
{replaced}%
{\Changes@replaced@id}%
{#2}%
{#3}%
{\Changes@replaced@comment}%
{\changesreplacedname}%
{#2}%
}
```

`\highlight` The command formats text as highlighted text.

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

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

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

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

```
\Changes@set@commandname{comment}
\expandafter\newcommand\expandafter{\csname\Changes@commandname\endcsname}[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}%
}
```

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`.

```
\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. change description with author
3. text
4. page

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

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

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

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

Default: two third of the text width

```
\newlength{\Changes@truncate@width}
\settruncatewidth{.6\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}
}
```

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

Default: one third of the text width

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

\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.

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

All work is done only in draft mode.

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

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

```
\IfIsInList{\Changes@loc@style}{list|summary|compactsummary}%
{}%
{%
\PackageWarning{changes}{Wrong style for list of changes: '\Changes@loc@style', us-
   ing 'list' instead.}%
\def\Changes@loc@style{}%
}%
\IfIsEmpty{\Changes@loc@style}%
{\def\Changes@loc@style{list}}%
{}%
```

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

```
\IfStrEq{\Changes@loc@show}{all}%
{\def\Changes@loc@show{added|deleted|replaced|highlight|comment}}%
{}%
\IfIsInList{\Changes@loc@show}{added|deleted|replaced|highlight|comment}%
{}%
{%
\PackageWarning{changes}{Wrong show-value for list of changes: '\Changes@loc@show', us-
   ing 'all' instead.}%
\def\Changes@loc@show{}%
}%
\IfIsEmpty{\Changes@loc@show}%
{\def\Changes@loc@show{added|deleted|replaced|highlight|comment}}%
{}%
```

Print heading.

```
\IfIsEmpty{\Changes@loc@title}%
{%
\IfStrEq{\Changes@loc@style}{list}%
{\def\Changes@heading{\listofchangesname}}{}%
\IfStrEq{\Changes@loc@style}{summary}%
{\def\Changes@heading{\summaryofchangesname}}{}%
\IfStrEq{\Changes@loc@style}{compactsummary}%
{\def\Changes@heading{\compactsummaryofchangesname}}{}%
}%
{\def\Changes@heading{\Changes@loc@title}}%
\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.

```
\IfIsInList{\Changes@loc@style}{list}%
{%
```

```
\IfFileExists{\jobname.\Changes@locextension}%
{%
\newread\Changes@InFile%
\openin\Changes@InFile=\jobname.\Changes@locextension%
\loop\unless\ifeof\Changes@InFile%
\read\Changes@InFile to \Changes@Line%
\ifeof\Changes@InFile\else%
\Changes@Line%
\fi
\repeat
\closein\Changes@InFile%
}{%
\emph{\changesnoloc}%
\PackageWarning{changes}{LaTeX rerun needed for list of changes}%
}%
}{}%
```

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.

```
\IfIsInList{\Changes@loc@style}{summary|compactsummary}%
{%
\IfFileExists{\jobname.\Changes@socextension}%
{%
\newread\Changes@InFile%
\openin\Changes@InFile = \jobname.\Changes@socextension%
```

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

```
\loop\unless\ifeof\Changes@InFile%
\read\Changes@InFile to \Changes@Line%
\ifeof\Changes@InFile\else%
\expandafter\changes@chopline\Changes@Line\\%
\textbf{%
\IfIsColored%
{\color{\Changes@Incolor}}%
{}%
\IfIsAnonymous{\Changes@Inid}%
{\changesauthorname: \changesanonymousname}%
{%
\changesauthorname: \Changes@Inid%
\IfIsEmpty{\Changes@Inname}%
{}%
{ (\Changes@Inname)}%
}%
}\\%
```

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

```
\numdef{\Changes@InSum}{0}%
\renewcommand*{\do}[1]{%
\numdef{\Changes@InSum}{\Changes@InSum + \csuse{Changes@In########1}}%
}%
\expandafter\dopsvlist\expandafter{\Changes@loc@show}%
```

Print summary.

```
\ifnumcomp{\Changes@InSum}{=}{0}%
{%
\parbox{\Changes@summary@width}{\changesnochanges}\\[1ex]%
}%
{%
\numdef{\Changes@InCount}{0}%
\renewcommand*{\do}[1]{%
\numdef{\Changes@InCount}{\Changes@InCount + \csuse{Changes@In########1}}%
\ifboolexpr{%
not test {\IfStrEq{\Changes@loc@style}{compactsummary}} or%
test {\ifnumgreater{\csuse{Changes@In########1}}{0}}%
}%
{%
\parbox{\Changes@summary@width}{\csuse{changes########1name}~\let\cleaders\leaders\dotfill~\
\ifnumless{\Changes@InCount}{\Changes@InSum}%
{\\}%
{\\[1ex]}%
}%
{}%
}%
\expandafter\dopsvlist\expandafter{\Changes@loc@show}%
}%
\fi%
```

File read loop.

```
\repeat
\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@socextension
```

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}
\loop\unless\ifnum\theChanges@Author = \value{Changes@AuthorCount}
\stepcounter{Changes@Author}%
\def\Changes@ID{\csuse{Changes@AuthorID\theChanges@Author}}%
\immediate\write\Changes@OutFile{\Changes@ID;%
\csuse{Changes@AuthorColor\Changes@ID};%
\csuse{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}}%
\repeat
```

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@\Changes@locextension}{
\expandafter\newwrite\csname tf@\Changes@locextension\endcsname
\immediate\openout \csname tf@\Changes@locextension\endcsname \jobname.\Changes@locextension
}{}
```

```
\fi
}
```

⟨/changes⟩