

# The *changes*-package

Manual change markup — version 4.0.4

May 13, 2021

**Ekkart Kleinod**

@ ekleinod@edgesoft.de

---

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Using the <i>changes</i>-package</b>	<b>5</b>
<b>3</b>	<b>Limitations and possible enhancements</b>	<b>8</b>
<b>4</b>	<b>User interface of the <i>changes</i>-package</b>	<b>9</b>
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	<code>\listofchanges</code>	20
4.5	Author management	21
4.5.1	<code>\definechangesauthor</code>	21
4.6	Adaptation of the output	21
4.6.1	Values for markup command definitions	22
4.6.2	<code>\setaddedmarkup</code>	22
4.6.3	<code>\setdeletedmarkup</code>	23
4.6.4	<code>\sethighlightmarkup</code>	23
4.6.5	<code>\setcommentmarkup</code>	24
4.6.6	<code>\setauthormarkup</code>	24
4.6.7	<code>\setauthormarkupposition</code>	25
4.6.8	<code>\setauthormarkuptext</code>	25
4.6.9	<code>\settruncatewidth</code>	25
4.6.10	<code>\setsummarywidth</code>	26
4.6.11	<code>\setsummarytewidth</code>	26
4.6.12	<code>\setlocextension</code>	26
4.6.13	<code>\setsoextension</code>	27
4.7	Used packages	27
<b>5</b>	<b>Remove markup from file</b>	<b>28</b>
<b>6</b>	<b>Known problems and solutions</b>	<b>29</b>
6.1	Special content	29
6.2	Footnotes and margin notes	29
6.3	The <i>ulem</i> package	29
6.4	Command already defined	30
<b>7</b>	<b>Authors</b>	<b>31</b>
<b>8</b>	<b>Versions</b>	<b>32</b>

---

<b>9</b>	<b>Distribution, Copyright, License</b>	<b>33</b>
<b>10</b>	<b>The documented sourcecode</b>	<b>34</b>
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	43
10.3	Packages	44
10.4	Language dependent texts	44
10.5	File extensions	47
10.6	Authors	47
10.6.1	Author management	47
10.6.2	Author markup	48
10.7	Change management commands	49
10.7.1	Text markup definition	49
10.7.2	Change management command definition	52
10.8	List of changes	59

## 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](mailto: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

This is **new** text. In this sentence, I replace a ~~good~~<sup>**bad**</sup> word. And, to sum up the text changes, there is another ~~obsolete~~<sup>**new**</sup> word to delete. Furthermore, text can be **highlighted**<sup>**new**</sup> or just commented.

[EK 2] For the fun of it.

Parallel to this manual is a folder “examples” which contains an extensive collection of example files, both  $\text{\LaTeX}$  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  $\text{\LaTeX}$
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 or comment 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  $\text{\LaTeX}$** 

After marking your changes in the text you are able to display them in the generated document by processing it as usual with  $\text{\LaTeX}$ . 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  $\TeX$  the data of the list is written into an auxiliary file. This data is used in the next  $\TeX$  run for typesetting the list of changes. Therefore, two  $\TeX$  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  $\TeX$  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 the most important known problems and possible solutions in Section 6. Please refer to the section first if your problem is known and 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](mailto:ekleinod@edgesoft.de). In that case, please start your email subject with [changes].

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

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

You can try putting such text in an extra file and include it 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:

<i>none</i>	no prefix, a command already defined raises an error (default strategy)
-------------	---

ifneeded	if a command is already defined, <i>changes</i> 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 <code>\comment</code> or <code>\highlight</code> are already defined and you mainly want to use the change commands <code>\added</code> , <code>\deleted</code> , and <code>\replaced</code> .
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:

<code>colored</code>	no text markup, just coloring – <b>example</b> (default)
<code>uline</code>	underlined text – <u>example</u>
<code>uuline</code>	double underlined text – <u><u>example</u></u>
<code>uwave</code>	wavy underlined text – <u>example</u>
<code>dashuline</code>	dashed underlined text – <u>example</u>
<code>dotuline</code>	dotted underlined text – <u>example</u>
<code>bf</code>	bold text – <b>example</b>
<code>it</code>	italic text – <i>example</i>
<code>sl</code>	slanted text – <i>example</i>
<code>em</code>	emphasized text – <i>example</i>

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 text. 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:

<code>sout</code>	striked out text – <del>example</del> (default)
<code>xout</code>	crossed out text – <del>example</del>
<code>colored</code>	no text markup, just coloring – <b>example</b>
<code>uline</code>	underlined text – <u>example</u>
<code>uuline</code>	double underlined text – <u><u>example</u></u>
<code>uwave</code>	wavy underlined text – <u>example</u>

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

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:

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

### Examples

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

#### 4.1.8 commentmarkup

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

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

The following values for *commentmarkup* are defined:

example  
comment

example  
comment

<code>todo</code>	comment as todo note, which is not added to list of todos (default)
<code>margin</code>	comment in margin
<code>footnote</code>	comment as footnote <sup>1</sup>
<code>uwave</code>	wavy underlined text – <u>example comment</u>

### 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:

<code>superscript</code>	superscripted text – text <sup>author</sup> (default)
<code>subscript</code>	subscripted text – text <sub>author</sub>
<code>brackets</code>	text in brackets – text(author)
<code>footnote</code>	text in footnote – text <sup>2</sup>
<code>none</code>	no author identification

### Examples

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

---

<sup>1</sup> example comment

<sup>2</sup> 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:

<code>right</code>	right of the text – $\text{text}^{\text{author}}$ (default)
<code>left</code>	left of the text – $\text{author}^{\text{text}}$

##### 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:

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

##### 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] anony-  
mous

This is new text. This is new<sup>EK</sup> 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

This is ~~old~~ text. This is ~~old~~<sup>EK</sup> text too. This is more ~~old~~ text. This is the last ~~old~~ text.

[2] away

### 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] bet-  
ter

This is newreplaced text. This is newreplaced<sup>EK</sup> text too. This is more newreplaced text. This is the last newreplaced text.

[3] im-  
proved

### 4.3 Highlighting and Comments

4.3.1 <code>\highlight</code>	19
4.3.2 <code>\comment</code>	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<sup>EK</sup> 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 `\comment` 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

This is commented text. This is commented text too.

[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  $\text{\TeX}$ -run creates an auxiliary file, the second run uses the data of this file. Therefore you need two  $\text{\TeX}$ -runs for an up-to-date list of changes.

There are three optional arguments:

<code>style</code>	list style
<code>title</code>	individual title
<code>show</code>	markup types

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

<code>list</code>	prints the list of changes like a list of figures (default)
<code>summary</code>	prints the number of changes grouped by author
<code>compactsummary</code>	same as <code>summary</code> 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:

<code>all</code>	show all types (default)
<code>added</code>	show only additions
<code>deleted</code>	show only deletions
<code>replaced</code>	show only replacements
<code>highlight</code>	show only highlights
<code>comment</code>	show only comments

## Examples

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

## 4.5 Author management

### 4.5.1 \definechangesauthor

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

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

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

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

The package predefines one anonymous author without id.

## Examples

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

## 4.6 Adaptation of the output

4.6.1	Values for markup command definitions	22
4.6.2	<code>\setaddedmarkup</code>	22
4.6.3	<code>\setdeletedmarkup</code>	23
4.6.4	<code>\sethighlightmarkup</code>	23
4.6.5	<code>\setcommentmarkup</code>	24

4.6.6	<code>\setauthormarkup</code>	24
4.6.7	<code>\setauthormarkupposition</code>	25
4.6.8	<code>\setauthormarkuptext</code>	25
4.6.9	<code>\settruncatewidth</code>	25
4.6.10	<code>\setsummarywidth</code>	26
4.6.11	<code>\setsummarytowidth</code>	26
4.6.12	<code>\setlocextension</code>	26
4.6.13	<code>\setsocextension</code>	27

#### 4.6.1 Values for markup command definitions

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

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

- any  $\LaTeX$ -commands
- author’s color can be used with color “authorcolor”
- boolean test if colored change output is needed “`\IfIsColored`”

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

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

#### 4.6.2 `\setaddedmarkup`

`\setaddedmarkup{<definition>}`

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

Values for definition:

- added text can be used with “#1”

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

### Examples

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

#### 4.6.3 \setdeletedmarkup

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

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

Values for definition:

- deleted text can be used with “#1”

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

### Examples

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

#### 4.6.4 \sethighlightmarkup

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

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

Values for definition:

- highlighted text can be used with “#1”

### Examples

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



#### 4.6.5 \setcommentmarkup

`\setcommentmarkup{<definition>}`

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

Values for definition:

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

##### Examples

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

#### 4.6.6 \setauthormarkup

`\setauthormarkup{<definition>}`

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

Values for definition:

- author output (id or name) can be used with “#1”

##### Examples

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

#### 4.6.7 `\setauthormarkupposition`

`\setauthormarkupposition{<authormarkupposition>}`

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

The following values for *authormarkupposition* are defined:

right	right of the text – text <sup>author</sup> (default)
left	left of the text – <sup>author</sup> text

#### Examples

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

#### 4.6.8 `\setauthormarkuptext`

`\setauthormarkuptext{<authormarkuptext>}`

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

The following values for *authormarkuptext* are defined:

id	author's id – text <sup>id</sup> (default)
name	author's name – text <sup>authorname</sup>

#### Examples

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

#### 4.6.9 `\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}
```

#### 4.6.10 `\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}
```

#### 4.6.11 `\setsummarytewidth`

```
\setsummarytewidth{<text>}
```

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

##### Examples

```
\setsummarytewidth{Highlighted \qqquad}  
\setsummarytewidth{The longest text imaginable for the summary.}
```

#### 4.6.12 `\setlocextension`

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

The command `\setlocextension` sets the extension of the auxiliary file for the list of changes (loc-file<sup>3</sup>). 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  $\TeX$  standard file extension, such as “toc” or “lof”, as this would collide with the normal  $\TeX$  run.

---

<sup>3</sup> “loc” stands for “list of changes”.

#### 4.6.13 `\setsoceextension`

`\setsoceextension{<extension>}`

The command `\setsoceextension` sets the extension of the auxiliary file for the summary of changes (soc-file<sup>4</sup>). 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

```
\setsoceextension{changes}  
\setsoceextension{chg}
```

Do not use a  $\TeX$  standard file extension, such as “toc” or “lof”, as this would collide with the normal  $\TeX$  run.

### 4.7 Used packages

The *changes*-package uses already existing packages for its 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 <code>\if</code> -commands, <i>bools</i> , 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)

---

<sup>4</sup> “soc” stands for “summary of changes”.

## 5 Remove markup from file

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

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

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

The script requires *python3*.

Use the script as follows:

```
python pyMergeChanges.py [-arh] <Input File> <Output File>
```

Options:

```
-a: accept all added, deleted and replaced  
-r: reject all added, deleted and replaced  
-h: remove all highlights
```

If no option is given, runs interactively.

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

```
python pyMergeChanges.py
```

Known issues:

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

## 6 Known problems and solutions

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

<https://gitlab.com/ekleinod/changes/issues>

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

### 6.1 Special content

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

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

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

### 6.2 Footnotes and margin notes

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

### 6.3 The *ulem* package

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

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

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

- Section 4.1.6
- Section 4.6.3

## 6.4 Command already defined

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

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

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

## 7 Authors

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

The authors are (in alphabetical order):

- Chiaradonna, Silvano
- Cui, Yvon
- Fischer, Ulrike
- Giovannini, Daniele
- Kleinod, Ekkart
- 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

Copyright 2007-2021 Ekkart Kleinod (ekleinod@edgesoft.de)

This work may be distributed and/or modified under the conditions of the  $\text{\LaTeX}$  Project Public License, either version 1.3 of this license or any later version. The latest version of this license is in <http://www.latex-project.org/lppl.txt> and version 1.3 or later is part of all distributions of  $\text{\LaTeX}$  version 2005/12/01 or later.

This work has the LPPL maintenance status “maintained”. The current maintainer of this work is Ekkart Kleinod.

This work consists of the files

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

and the derived files

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

## 10 The documented sourcecode

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

```
1 <*changes>
```

### 10.1 Package information and options

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

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{changes}[2021/05/13 v4.0.4 changes package]
4 \typeout{*** changes package 2021/05/13 v4.0.4 ***}
```

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

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

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

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

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

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

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

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

Declare list parsers.

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

A temporary list and flag.

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

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

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

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

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

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

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

End fo the workaround for `\forpsvlist`.

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

### 10.1.1 Package options

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

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

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

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

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

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

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

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

Option markup, sets markup options accordingly.

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

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

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

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

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

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

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

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

118 }

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

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

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

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

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

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

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

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

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

Store default author color..

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

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

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

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

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

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

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

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

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



Unknown options generate a package warning.

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

### 10.1.2 Command options

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

`\definechangesauthor`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

`\added`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

### `\deleted`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

### `\replaced`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

### `\highlight`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

`\comment`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

`\listofchanges`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

### 10.1.3 Package options

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

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

### 10.1.4 Option processing

Process the options.

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

## 10.2 Utility tests

`\IfIsColored` Check if text should be colored.

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

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

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

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

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

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

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

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

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

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

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

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

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

### 10.3 Packages

Package *xcolor* provides colored text.

```
263 \IfIsColored
264 {\RequirePackage{xcolor}}
265 {}
```

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

```
266 \ifboolexpr{
267 test {\IfIsInList{\Changes@optionaddedmarkup}{dashuline/dotuline/sout/uline/uuline/uwave/xou
268 test {\IfIsInList{\Changes@optiondeletedmarkup}{dashuline/dotuline/sout/uline/uuline/uwave/x
269 test {\IfIsInList{\Changes@optionhighlightmarkup}{dashuline/dotuline/sout/uline/uuline/uwave/x
270 }
271 {\RequirePackage{ulem}}
272 {}
```

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

```
273 \IfStrEq{\Changes@optioncommentmarkup}{todo}
274 {\RequirePackage{todonotes}}
275 {}
```

### 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.

```
276 \def\listofchangesname{List of changes}
277 \def\summaryofchangesname{Changes}
278 \def\compactsummaryofchangesname{Changes (compact)}
279 \def\changesaddedname{Added}
```

```
280 \def\changesdeletedname{Deleted}
281 \def\changesreplacedname{Replaced}
282 \def\changeshighlightname{Highlighted}
283 \def\changescommentname{Commented}
284 \def\changesauthorname{Author}
285 \def\changesanonymousname{anonymous}
286 \def\changesnochanges{No changes.}
287 \def\changesnoloc{List of changes is available after the next \LaTeX\ run.}
288 \def\changesnosoc{Summary of changes is available after the next \LaTeX\ run.}
```

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

```
289 \ifboolexpr{
290 test {\@ifpackageloaded{babel}} or
291 test {\@ifpackageloaded{polyglossia}}
292 }
293 {
294 \addto\captionsngerman{\def\listofchangesname{Liste der \\"Anderungen}}
295 \addto\captionsngerman{\def\summaryofchangesname{\\"Anderungen}}
296 \addto\captionsngerman{\def\compactsummaryofchangesname{\\"Anderungen (kompakt)}}
297 \addto\captionsngerman{\def\changesaddedname{Eingef\\\"ugt}}
298 \addto\captionsngerman{\def\changesdeletedname{Gel\\\"oscht}}
299 \addto\captionsngerman{\def\changesreplacedname{Ersetzt}}
300 \addto\captionsngerman{\def\changeshighlightname{Hervorgehoben}}
301 \addto\captionsngerman{\def\changescommentname{Kommentiert}}
302 \addto\captionsngerman{\def\changesauthorname{Autor:in}}
303 \addto\captionsngerman{\def\changesanonymousname{Anonym}}
304 \addto\captionsngerman{\def\changesnochanges{Keine \\"Anderungen.}}
305 \addto\captionsngerman{\def\changesnoloc{Liste der \\"Anderungen nach dem n\\\"achsten \La-
TeX-Lauf verf\\\"ugbar.}}
306 \addto\captionsngerman{\def\changesnosoc{\\"Anderungen nach dem n\\\"achsten \La-
TeX-Lauf verf\\\"ugbar.}}
307
308 \addto\captionsngerman{\def\listofchangesname{Liste der \\"Anderungen}}
309 \addto\captionsngerman{\def\summaryofchangesname{\\"Anderungen}}
310 \addto\captionsngerman{\def\compactsummaryofchangesname{\\"Anderungen (kompakt)}}
311 \addto\captionsngerman{\def\changesaddedname{Eingef\\\"ugt}}
312 \addto\captionsngerman{\def\changesdeletedname{Gel\\\"oscht}}
313 \addto\captionsngerman{\def\changesreplacedname{Ersetzt}}
314 \addto\captionsngerman{\def\changeshighlightname{Hervorgehoben}}
315 \addto\captionsngerman{\def\changescommentname{Kommentiert}}
316 \addto\captionsngerman{\def\changesauthorname{Autor:in}}
317 \addto\captionsngerman{\def\changesanonymousname{Anonym}}
318 \addto\captionsngerman{\def\changesnochanges{Keine \\"Anderungen.}}
319 \addto\captionsngerman{\def\changesnoloc{Liste der \\"Anderungen nach dem n\\\"achsten \La-
TeX-Lauf verf\\\"ugbar.}}
320 \addto\captionsngerman{\def\changesnosoc{\\"Anderungen nach dem n\\\"achsten \La-
TeX-Lauf verf\\\"ugbar.}}
321
322 \addto\captionsenglish{\def\listofchangesname{List of changes}}
```

```
323 \addto\captionsenglish{\def\summaryofchangesname{Changes}}
324 \addto\captionsenglish{\def\compactsummaryofchangesname{Changes (compact)}}
325 \addto\captionsenglish{\def\changesaddedname{Added}}
326 \addto\captionsenglish{\def\changesdeletedname{Deleted}}
327 \addto\captionsenglish{\def\changesreplacedname{Replaced}}
328 \addto\captionsenglish{\def\changeshighlightname{Highlighted}}
329 \addto\captionsenglish{\def\changescommentname{Commented}}
330 \addto\captionsenglish{\def\changesauthorname{Author}}
331 \addto\captionsenglish{\def\changesanonymousname{anonymous}}
332 \addto\captionsenglish{\def\changesnochanges{No changes.}}
333 \addto\captionsenglish{\def\changesnoloc{List of changes is available after the next \La-
    TeX\ run.}}
334 \addto\captionsenglish{\def\changesnosoc{Summary of changes is available af-
    ter the next \LaTeX\ run.}}
335
336 \addto\captionsenglish{\def\listofchangesname{List of changes}}
337 \addto\captionsenglish{\def\summaryofchangesname{Changes}}
338 \addto\captionsenglish{\def\compactsummaryofchangesname{Changes (compact)}}
339 \addto\captionsenglish{\def\changesaddedname{Added}}
340 \addto\captionsenglish{\def\changesdeletedname{Deleted}}
341 \addto\captionsenglish{\def\changesreplacedname{Replaced}}
342 \addto\captionsenglish{\def\changeshighlightname{Highlighted}}
343 \addto\captionsenglish{\def\changescommentname{Commented}}
344 \addto\captionsenglish{\def\changesauthorname{Author}}
345 \addto\captionsenglish{\def\changesanonymousname{anonymous}}
346 \addto\captionsenglish{\def\changesnochanges{No changes.}}
347 \addto\captionsenglish{\def\changesnoloc{List of changes is available after the next \La-
    TeX\ run.}}
348 \addto\captionsenglish{\def\changesnosoc{Summary of changes is available af-
    ter the next \LaTeX\ run.}}
349
350 \addto\captionsenglish{\def\listofchangesname{Lista delle modifiche}}
351 \addto\captionsenglish{\def\summaryofchangesname{Modifiche}}
352 \addto\captionsenglish{\def\compactsummaryofchangesname{Modifiche (coerente)}} % trans-
    lation by me (EK), please provide correct translation
353 \addto\captionsenglish{\def\changesaddedname{Aggiunte}}
354 \addto\captionsenglish{\def\changesdeletedname{Cancellazioni}}
355 \addto\captionsenglish{\def\changesreplacedname{Sostituzioni}}
356 \addto\captionsenglish{\def\changeshighlightname{Accentare}} % translation by me (EK), please
    provide correct translation
357 \addto\captionsenglish{\def\changescommentname{Commenti}} % translation by me (EK), please
    provide correct translation
358 \addto\captionsenglish{\def\changesauthorname{Autore}}
359 \addto\captionsenglish{\def\changesanonymousname{anonimo}}
360 \addto\captionsenglish{\def\changesnochanges{Nessuna modifica.}} % translation by me (EK), please
    provide correct translation
361 \addto\captionsenglish{\def\changesnoloc{La lista delle modifiche sar\'a disponi-
    bile alla prossima esecuzione di \LaTeX.}}
362 \addto\captionsenglish{\def\changesnosoc{Le modifiche sar\'a disponibile alla prossima es-
```

```
    ecuzione di \LaTeX.}}  
363 }  
364 {}
```

## 10.5 File extensions

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

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

`setlocextension` Sets a new file extension for list of changes.

```
366 \newcommand{\setlocextension}[1]{  
367 \renewcommand{\Changes@locextension}{#1}  
368 }
```

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

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

`setsocextension` Sets a new file extension for summary of changes.

```
370 \newcommand{\setsocextension}[1]{  
371 \renewcommand{\Changes@socextension}{#1}  
372 }
```

## 10.6 Authors

### 10.6.1 Author management

Author counter.

```
373 \newcounter{Changes@AuthorCount}  
374 \setcounter{Changes@AuthorCount}{0}  
375 \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.

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



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

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

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

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

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

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

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

```
380 \expandafter\let\csname Changes@AuthorName#2\endcsname=\Changes@definechangesauthor@name
381 \expandafter\let\csname Changes@AuthorColor#2\endcsname=\Changes@definechangesauthor@color
382 \newcounter{Changes@addedCount#2}
383 \newcounter{Changes@deletedCount#2}
384 \newcounter{Changes@replacedCount#2}
385 \newcounter{Changes@highlightCount#2}
386 \newcounter{Changes@commentCount#2}
387 }
```

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

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

### 10.6.2 Author markup

`\Markup@author` Store markup for authors.

```
389 \newcommand{\Changes@Markup@author}[1]{%
390 \IfStrEq{\Changes@optionauthormarkup}{superscript}%
391 {\textsuperscript{#1}}%
392 }%
393 \IfStrEq{\Changes@optionauthormarkup}{subscript}%
394 {\textsubscript{#1}}%
395 }%
396 \IfStrEq{\Changes@optionauthormarkup}{brackets}%
397 {(#1)}%
398 }%
399 \IfStrEq{\Changes@optionauthormarkup}{footnote}%
400 {\footnote{#1}}%
401 }%
402 \IfStrEq{\Changes@optionauthormarkup}{none}%
403 }%
404 }%
405 }
```

`\setauthormarkup` Set markup for authors.

```
406 \newcommand{\setauthormarkup}[1]{  
407 \renewcommand{\Changes@Markup@author}[1]{#1}  
408 }
```

`\markupposition` Set position for author markup text.

```
409 \newcommand{\setauthormarkupposition}[1]{  
410 \renewcommand{\Changes@optionauthormarkupposition}{#1}  
411 }
```

`\authormarkuptext` Set author markup text to be displayed.

```
412 \newcommand{\setauthormarkuptext}[1]{  
413 \renewcommand{\Changes@optionauthormarkuptext}{#1}  
414 }
```

## 10.7 Change management commands

### 10.7.1 Text markup definition

Replaced text is always typeset as follows:  $\langle added\ text \rangle \langle deleted\ text \rangle$ . Therefore no extra command for markup of replaced text is given.

`\Changes@Markup` Predefined markups for text.

```
415 \newcommand{\Changes@Markup}[2]{%  
416 \IfStrEq{#1}{background}%  
417 {%  
418 \IfIsColored%  
419 {\colorbox{authorcolor!30}{#2}}%  
420 {#2}}%  
421 }%  
422 {%}  
423 \IfStrEq{#1}{bf}%  
424 {\textbf{#2}}%  
425 {%}  
426 \IfStrEq{#1}{colored}%  
427 {#2}%  
428 {%}  
429 \IfStrEq{#1}{dashuline}%  
430 {\dashuline{#2}}%  
431 {%}  
432 \IfStrEq{#1}{dotuline}%  
433 {\dotuline{#2}}%  
434 {%}
```

```
435 \IfStrEq{#1}{em}%
436 {\emph{#2}}%
437 {}%
438 \IfStrEq{#1}{it}%
439 {\textit{#2}}%
440 {}%
441 \IfStrEq{#1}{sl}%
442 {\textsl{#2}}%
443 {}%
444 \IfStrEq{#1}{sout}%
445 {\sout{#2}}%
446 {}%
447 \IfStrEq{#1}{uline}%
448 {\uline{#2}}%
449 {}%
450 \IfStrEq{#1}{uuline}%
451 {\uuline{#2}}%
452 {}%
453 \IfStrEq{#1}{uwave}%
454 {\uwave{#2}}%
455 {}%
456 \IfStrEq{#1}{xout}%
457 {\xout{#2}}%
458 {}%
459 }
```

es@Markup@added Store markup for added text.

```
460 \newcommand{\Changes@Markup@added}[1]{%
461 \Changes@Markup{\Changes@optionaddedmarkup}{#1}}%
462 }
```

setaddedmarkup Set markup for added text.

```
463 \newcommand{\setaddedmarkup}[1]{
464 \renewcommand{\Changes@Markup@added}[1]{#1}
465 }
```

@Markup@deleted Store markup for deleted text.

```
466 \newcommand{\Changes@Markup@deleted}[1]{%
467 \Changes@Markup{\Changes@optiondeletedmarkup}{#1}}%
468 }
```

etdeletedmarkup Set markup for deleted text.

```
469 \newcommand{\setdeletedmarkup}[1]{
470 \renewcommand{\Changes@Markup@deleted}[1]{#1}
471 }
```

Markup@highlight Store markup for highlighted text.

```
472 \newcommand{\Changes@Markup@highlight}[1]{%
473 \Changes@Markup{\Changes@optionhighlightmarkup}{#1}}%
474 }
```

highlightmarkup Set markup for highlighted text.

```
475 \newcommand{\sethighlightmarkup}[1]{
476 \renewcommand{\Changes@Markup@highlight}[1]{#1}
477 }
```

Markup@comment Store markup for comments.

Parameters:

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

```
478 \newcommand{\Changes@Markup@comment}[3]{%
479 \IfStrEq{\Changes@optioncommentmarkup}{todo}%
480 {%
481 \IfIsColored%
482 {\colorlet{\Changes@todocolor}{authorcolor}}%
483 {\colorlet{\Changes@todocolor}{black}}%
484 \todo[color=\Changes@todocolor!10, bordercolor=\Changes@todocolor, linecolor=\Changes@todocolor]
    list]{\textbf{[\IfIsAnonymous{#2}]{#3~}\arabic{authorcommentcount}} #1}%
485 }%
486 \IfStrEq{\Changes@optioncommentmarkup}{margin}%
487 {%
488 \marginpar{%
489 \IfIsColored%
490 {\leavevmode\color{authorcolor}}%
491 }%
492 \textbf{[\IfIsAnonymous{#2}]{#3~}\arabic{Changes@commentCount#2}]:} #1%
493 }%
494 }%
495 \IfStrEq{\Changes@optioncommentmarkup}{footnote}%
496 {%
497 \footnote{%
498 \textbf{[\IfIsAnonymous{#2}]{#3~}\arabic{Changes@commentCount#2}]:} #1%
499 }%
500 }%
501 \IfStrEq{\Changes@optioncommentmarkup}{uwave}%
502 {%
503 {%
504 \IfIsColored%
505 {\color{authorcolor}}%
```

```
506 {}%
507 \allowbreak%
508 \uwave{%
509 \textbf{[\IfIsAnonymous{#2}{}{#3~}\arabic{Changes@commentCount#2}]:} #1%
510 }%
511 }%
512 }{}%
513 }
```

etcommentmarkup Set markup for comments.

```
514 \newcommand{\setcommentmarkup}[1]{
515 \renewcommand{\Changes@Markup@comment}[3]{#1}
516 }
```

### 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

```
517 \newbool{Changes@WrongID}
518 \newcommand{\Changes@check@author}[1]{%
519 \IfIsEmpty{#1}%
520 {}%
521 {%
522 \setbool{Changes@WrongID}{true}%
523 \setcounter{Changes@Author}{0}%
524 \loop%
525 \stepcounter{Changes@Author}%
526 \IfStrEq{#1}{\csuse{Changes@AuthorID\theChanges@Author}}%
527 {\setbool{Changes@WrongID}{false}}%
528 {}%
529 \ifnum\theChanges@Author<\theChanges@AuthorCount%
530 \repeat%
531 \ifbool{Changes@WrongID}%
532 {%
533 \PackageError{changes}%
534 {Undefined changes author: #1}%
535 {You have to define the author #1 with e.g.: \definechangesauthor{#1}}%
536 }%
537 {}%
538 }
```

539 }

`\changes@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.

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

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

```
541 \IfIsAuthorEmptyAtPosition{#1}{#2}%  
542 {}%  
543 {%  
544 \IfStrEq{\Changes@optionauthormarkuptext}{id}%  
545 {#1}%  
546 {}%  
547 \IfStrEq{\Changes@optionauthormarkuptext}{name}%  
548 {%
```

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

```
549 \IfStrEq{\csuse{Changes@AuthorName#1}}{}%  
550 {#1}%  
551 {\csuse{Changes@AuthorName#1}}%  
552 }%  
553 {}%  
554 }%  
555 }
```

`\changes@set@color` Sets the author's color.

This command has the following argument:

1. author's id

```
556 \newrobustcmd{\Changes@set@color}[1]{%  
557 \IfIsColored%  
558 {\colorlet{authorcolor}{\csuse{Changes@AuthorColor#1}}}%  
559 {}%  
560 }
```

`et@commentcount` Sets the author's comment count.

This command has the following argument:

1. author's id

```
561 \newcounter{authorcommentcount}
562 \newrobustcmd{\Changes@set@commentcount}[1]{%
563 \setcounter{authorcommentcount}{\value{Changes@commentCount#1}}}%
564 }
```

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

```
565 \def\Changes@commandprefix{ch}
566 \newrobustcmd{\Changes@set@commandname}[1]{
567 \def\Changes@commandname{#1}
568 \IfStrEq{\Changes@optioncommandnameprefix}{always}
569 {
570 \def\Changes@commandname{\Changes@commandprefix#1}
571 \typeout{changes-package option commandnameprefix=always: using prefix on #1, new com-
572 }{mand name is \Changes@commandname.}
573 \ifcsdef{#1}
574 {
575 \IfStrEq{\Changes@optioncommandnameprefix}{none}
576 {
577 \PackageError{changes}
578 {Command #1 is already defined.}
579 {Try package option commandnameprefix with "always" or "ifneeded", e.g. usep-
580 }{}ackage[commandnameprefix=always]{changes}}
581 \IfStrEq{\Changes@optioncommandnameprefix}{ifneeded}
582 {
583 \def\Changes@commandname{\Changes@commandprefix#1}
584 \PackageWarning{changes}{Command #1 is already defined, using \Changes@commandname}
585 }{}
586 }
587 {}
588 }
589 }
```

`\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

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

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

```
591 \ifbool{Changes@optiondraft}%  
592 {%
```

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

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

Start output.

```
595 {%
```

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

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

```
596 \IfIsInList{#1}{added/deleted/replaced/highlight}%  
597 {%
```

Author text for left positioning (only without comment).

```
598 \IfIsEmpty{#5}%  
599 {%  
600 \IfIsAuthorEmptyAtPosition{#2}{left}%  
601 }}%  
602 {}%  
603 \IfIsColored%  
604 {\color{authorcolor}}%  
605 }}%  
606 \Changes@Markup@author{\Changes@output@author{#2}{left}}%  
607 }}%  
608 }{}%
```

Changed/highlighted text.

```
609 {%  
610 \IfStrEq{#1}{highlight}%  
611 }{}%  
612 \IfIsColored%  
613 {\color{authorcolor}}%  
614 }}%  
615 }%  
616 \IfStrEq{#1}{added}{\Changes@Markup@added{#3}}{}%
```



```
617 \IfStrEq{#1}{deleted}{\Changes@Markup@deleted{#4}}{}%
618 \IfStrEq{#1}{replaced}{\Changes@Markup@added{#3}}\allowbreak\Changes@Markup@deleted{#4}}{}%
619 \IfStrEq{#1}{highlight}{\Changes@Markup@highlight{#3}}{}%
620 }%
```

Author text for right positioning (only without comment).

```
621 \IfIsEmpty{#5}%
622 {%
623 \IfIsAuthorEmptyAtPosition{#2}{right}%
624 }%
625 {%
626 \IfIsColored%
627 {\color{authorcolor}}%
628 }%
629 \Changes@Markup@author{\Changes@output@author{#2}{right}}%
630 }%
631 }%
```

Update counters.

```
632 \stepcounter{Changes@#1Count#2}%
633 }%
```

Comments.

```
634 \IfIsEmpty{#5}%
635 }%
636 {%
637 \stepcounter{Changes@commentCount#2}%
638 \Changes@set@commentcount{#2}%
639 \Changes@Markup@comment%
640 {#5}%
641 {#2}%
642 {\Changes@output@author{#2}{left}\Changes@output@author{#2}{right}}%
643 }%
644 }%
```

Store line for list of changes.

```
645 \IfIsEmpty{#2}%
646 {\def\Changes@locid{}}%
647 {\def\Changes@locid{~(#{2})}}%
648 \addtocontents{\Changes@locextension}{\protect\ChangesListline{#1}{#6\Changes@locid}{#7}{\th
649 }%
```

Output unchanged text (option final was set).

```
650 {%
651 \IfIsEmpty{#3}%
652 {\@bsphack\@esphack}%
```

```
653 {#3}%  
654 }%  
655 }
```

`\added` The command formats text as new text.

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

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

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

```
658 \setkeys{Changes@added}{#1}%  
659 \Changes@output%  
660 {added}%  
661 {\Changes@added@id}%  
662 {#2}%  
663 }%  
664 {\Changes@added@comment}%  
665 {\changesaddedname}%  
666 {#2}%  
667 }
```

`\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

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

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

```
670 \setkeys{Changes@deleted}{#1}%  
671 \Changes@output%  
672 {deleted}%  
673 {\Changes@deleted@id}%  
674 }%  
675 {#2}%  
676 {\Changes@deleted@comment}%  
677 {\changesdeletedname}%  
678 {#2}%  
679 }
```

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

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

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

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

```
682 \setkeys{Changes@replaced}{#1}%  
683 \Changes@output%  
684 {replaced}%  
685 {\Changes@replaced@id}%  
686 {#2}%  
687 {#3}%  
688 {\Changes@replaced@comment}%  
689 {\changesreplacedname}%  
690 {#2}%  
691 }
```

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

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

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

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

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

`\comment` The command formats text as comment.

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

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

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

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

## 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`.

```
716 \RequirePackage{truncate}  
717 \renewcommand\TruncateMarker{ [\dots\negthinspace]\ }
```

`Changes@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.

```
718 \def\changes@chopline#1;#2;#3;#4;#5;#6;#7;#8 \{\%  
719 \def\Changes@Inid{#1}%  
720 \def\Changes@Incolor{#2}%  
721 \def\Changes@Inname{#3}%  
722 \def\Changes@Inadded{#4}%  
723 \def\Changes@Indeleted{#5}%  
724 \def\Changes@Inreplaced{#6}%  
725 \def\Changes@Inhighlight{#7}%  
726 \def\Changes@Incomment{#8}%  
727 }
```

`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

```
728 \newcommand{\ChangesListline}[4]{%
729 \IfIsInList{#1}{\Changes@loc@show}{%
730 \@ifundefined{@dotsep}%
731 {\def\@dotsep{4.5}}%
732 }%
733 \@dottedtocline{1}{0pt}{2em}{#2: \truncate{\Changes@truncate@width}{#3}}{#4}%
734 }%
735 }
```

`\settruncatewidth` Set the width of the truncation. Argument: new width.

```
736 \newcommand{\settruncatewidth}[1]{
737 \setlength{\Changes@truncate@width}{#1}
738 }
```

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

Default: two third of the text width

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

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

```
741 \newcommand{\setsummarywidth}[1]{
742 \setlength{\Changes@summary@width}{#1}
743 }
```

`\@summarytewidth` Set the width of the change summary to width of given text. Argument: text.

```
744 \newcommand{\setsummarytewidth}[1]{
745 \settowidth{\Changes@summary@width}{#1}
746 }
```

`\@summary@width` Length for the width of the change summary.

Default: one third of the text width

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

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

The following styles are available:

<code>list</code>	prints the list of changes like a list of figures
<code>summary</code>	prints the number of changes grouped by author
<code>compactsummary</code>	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.

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

All work is done only in draft mode.

```
751 \ifbool{Changes@optiondraft}%  
752 {%
```

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

```
753 \IfIsInList{\Changes@loc@style}{list/summary/compactsummary}%  
754 {}%  
755 {%  
756 \PackageWarning{changes}{Wrong style for list of changes: '\Changes@loc@style', us-  
ing 'list' instead.}%  
757 \def\Changes@loc@style{}%  
758 }%  
759 \IfIsEmpty{\Changes@loc@style}%  
760 {\def\Changes@loc@style{list}}%  
761 {}%
```

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

```
762 \IfStrEq{\Changes@loc@show}{all}%  
763 {\def\Changes@loc@show{added|deleted|replaced|highlight|comment}}%  
764 {}%  
765 \IfIsInList{\Changes@loc@show}{added|deleted|replaced|highlight|comment}%  
766 {}%  
767 {%  
768 \PackageWarning{changes}{Wrong show-value for list of changes: '\Changes@loc@show', us-  
ing 'all' instead.}%  
769 \def\Changes@loc@show{}%  
770 }%  
771 \IfIsEmpty{\Changes@loc@show}%  
772 {\def\Changes@loc@show{added|deleted|replaced|highlight|comment}}%  
773 {}%
```

Print heading.

```
774 \IfIsEmpty{\Changes@loc@title}%  
775 {%  
776 \IfStrEq{\Changes@loc@style}{list}%
```

```
777 {\def\Changes@heading{\listofchangesname}}}%
778 \IfStrEq{\Changes@loc@style}{summary}%
779 {\def\Changes@heading{\summaryofchangesname}}}%
780 \IfStrEq{\Changes@loc@style}{compactsummary}%
781 {\def\Changes@heading{\compactsummaryofchangesname}}}%
782 }%
783 {\def\Changes@heading{\Changes@loc@title}}%
784 \section*{\Changes@heading}
```

Print list.

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

```
785 \IfIsInList{\Changes@loc@style}{list}%
786 {%
787 \IfFileExists{\jobname.\Changes@locextension}%
788 {%
789 \newread\Changes@InFile%
790 \openin\Changes@InFile=\jobname.\Changes@locextension%
791 \loop\unless\ifeof\Changes@InFile%
792 \read\Changes@InFile to \Changes@Line%
793 \ifeof\Changes@InFile\else%
794 \Changes@Line%
795 \fi
796 \repeat
797 \closein\Changes@InFile%
798 }%
799 \emph{\changesnoloc}%
800 \PackageWarning{changes}{LaTeX rerun needed for list of changes}%
801 }%
802 }%}
```

Print summary or compact summary.

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

```
803 \IfIsInList{\Changes@loc@style}{summary|compactsummary}%
804 {%
805 \IfFileExists{\jobname.\Changes@socextension}%
806 {%
807 \newread\Changes@InFile%
808 \openin\Changes@InFile = \jobname.\Changes@socextension%
```

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

```
809 \loop\unless\ifeof\Changes@InFile%
810 \read\Changes@InFile to \Changes@Line%
811 \ifeof\Changes@InFile\else%
```

```
812 \expandafter\changes@chopline\Changes@Line\%
813 \textbf{%
814 \IfIsColored%
815 {\color{\Changes@Incolor}}%
816 }%
817 \IfIsAnonymous{\Changes@Inid}%
818 {\changesauthorname: \changesanonymousname}%
819 {%
820 \changesauthorname: \Changes@Inid%
821 \IfIsEmpty{\Changes@Inname}%
822 }%
823 { (\Changes@Inname)}%
824 }%
825 } \%
```

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

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

```
826 \numdef{\Changes@InSum}{0}%
827 \def\do{%
828 \renewcommand*{\do}[1]{%
829 \numdef{\Changes@InSum}{\Changes@InSum + \csuse{\Changes@In#####1}}%
830 }%
831 \expandafter\dopsvlist\expandafter{\Changes@loc@show}%
```

Print summary.

```
832 \ifnumcomp{\Changes@InSum}{=}{0}%
833 {%
834 \parbox{\Changes@summary@width}{\changesnochanges}\[1ex]%
835 }%
836 {%
837 \numdef{\Changes@InCount}{0}%
838 \renewcommand*{\do}[1]{%
839 \numdef{\Changes@InCount}{\Changes@InCount + \csuse{\Changes@In#####1}}%
840 \ifboolexpr{%
841 not test {\IfStrEq{\Changes@loc@style}{compactsummary}} or%
842 test {\ifnumgreater{\csuse{\Changes@In#####1}}{0}}%
843 }%
844 {%
845 \parbox{\Changes@summary@width}{\csuse{changes#####1name}~\let\cleaders\leaders\dotfill~}
846 \ifnumless{\Changes@InCount}{\Changes@InSum}%
847 {\%
848 {\[1ex]}%
849 }%
850 }%
851 }%
852 \expandafter\dopsvlist\expandafter{\Changes@loc@show}%
```



```
853 }%  
854 \fi%
```

File read loop.

```
855 \repeat  
856 \closein\Changes@InFile%  
857 }{%  
858 \emph{\changesnosoc}%  
859 \PackageWarning{changes}{LaTeX rerun needed for summary of changes}%  
860 }%  
861 }{}%
```

In final mode print nothing.

```
862 }{}%  
863 }
```

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.

```
864 \AtEndDocument{%
```

Open output file.

```
865 \newwrite\Changes@OutFile  
866 \immediate\openout\Changes@OutFile = \jobname.\Changes@socextension
```

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

```
867 \let\Changes@protect\protect  
868 \let\protect\@unexpandable@protect
```

Output data for list of changes.

```
869 \setcounter{Changes@Author}{0}  
870 \loop\unless\ifnum\theChanges@Author = \value{Changes@AuthorCount}  
871 \stepcounter{Changes@Author}%  
872 \def\Changes@ID{\csuse{Changes@AuthorID}\theChanges@Author}}%  
873 \immediate\write\Changes@OutFile{\Changes@ID;%  
874 \csuse{Changes@AuthorColor}\Changes@ID;%  
875 \csuse{Changes@AuthorName}\Changes@ID;%  
876 \the\value{Changes@addedCount}\Changes@ID;%  
877 \the\value{Changes@deletedCount}\Changes@ID;%  
878 \the\value{Changes@replacedCount}\Changes@ID};%
```

```
879 \the\value{Changes@highlightCount\Changes@ID};%  
880 \the\value{Changes@commentCount\Changes@ID}}%  
881 \repeat
```

Close output file.

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

Restore original definition of \protect.

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

Write content of listofchanges to file.

```
884 \if@files  
885 \@ifundefined{tf@\Changes@locextension}{  
886 \expandafter\newwrite\csname tf@\Changes@locextension\endcsname  
887 \immediate\openout \csname tf@\Changes@locextension\endcsname \jobname.\Changes@locextension  
888 }{}  
889 \fi  
890 }
```

```
891 \</changes>
```