The *changes*-package

Manual change markup — version 2.1.0

October 10, 2018

Ekkart Kleinod

✉ ekleinod@edgesoft.de

Contents

1 Introduction

This package provides means for manual change markup.

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

`http://edgesoft.de/projects/latex/changes/`

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

> README: 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. The package allows free definition of additional authors and their associated color. It also allows you to change the markup of changes, authors, or annotations.

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. In order to use the package, you have to follow these steps:

1. use *changes*-package
2. define authors
3. markup text changes
4. typeset the document with LaTeX
5. output list of changes
6. 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 4.4.

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[<options>]{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 4.3.

markup text changes

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

for newly added text:
`\added[id=<id>, remark=<remark>]{new text}`

for deleted text:
`\deleted[id=<id>, remark=<remark>]{old text}`

for replaced text:
`\replaced[id=<id>, remark=<remark>]{⟨new text⟩}{⟨old text⟩}`

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

For detailed information please refer to Section 4.2.

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=<list|summary>]`

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`.

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.

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}`

2.1 Available scripts

In order to remove the markup from the LaTeX source code you can use a script from Silvano Chiaradonna. You find the script in the directory:

`<texpath>/scripts/changes/`

The script removes all markups. You can select or deselect markup from removal using the interactive mode. Switch on the interactive mode with the script parameter `-i`.

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 functionionality. In that case, please go to

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

There you can report errors, ask for help in the forum, or give advice to other users. You can view the source code, and change it according to your needs. I will try to include your changes in the maintained package. If you are a registered *sourceforge* user you can be a co-author of the *changes*-package.

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 can markup more than one paragraph at a time but occasionally this leads to errors. You cannot markup figures or tables.

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.

There is a problem of typesetting footnotes in special environments, such as tables or tabbings. Since footnotes are the default markup of remarks, this would be a problem. You can solve this problem by defining another annotation of remarks.

There are several possibilities of enhancing the *changes*-package. I will describe but a few here, I will not implement them due to lack of time and/or skill. You can have a look at the more complete list of enhancements on the *sourceforge* page.

- selecting of acknowledged and rejected texts; deletion of the corresponding markup
- markup of more than one paragraph
- markup of figures and tables
- automatic markup based on diff information (with regard to the limitations, such as markup of paragraphs, figures etc.)
- translation of language dependent texts and the user documentation in other languages

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 respectively 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.

A preliminary remark regarding typesetting of replaced text: replaced text is always typeset as follows: ⟨*new text*⟩⟨*old text*⟩. Thus, there is no possiblity to influence the output of replaced text directly, but via changing the output of added respectively deleted text.

4.1 Package Options

4.1.1 draft

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`.

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

4.1.2 final

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`.

`\usepackage[final]{changes}`

4.1.3 markup

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` and/or `deletedmarkup`.

The following values are allowed:

`default`
: colored markup of added text, striked out for deleted text (default markup)

`underlined`
: underlined for added text, striked out for deleted text

`bfit`
: bold added text, italic deleted text

`nocolor`
: no colored markup, underlined for added text, striked out for deleted text

Call
`\usepackage[markup=`⟨*markup*⟩`]{changes}`
Examples
`\usepackage[markup=default]{changes}` ≙ `\usepackage{changes}`
`\usepackage[markup=underlined]{changes}`
`\usepackage[markup=bfit]{changes}`
`\usepackage[markup=nocolor]{changes}`

4.1.4 addedmarkup, deletedmarkup

The `addedmarkup` option chooses a predefined visual markup of added text. The `deletedmarkup` option chooses a predefined visual markup of deleted text respectively. The default markup is chosen if no explicit markup is given. The options `addedmarkup` and `deletedmarkup` overwrite the markup chosen with `markup`.

The following values are allowed:

`none`
: no markup – example (default markup for added text)

`uline`
: underlined text – example

`uuline`
: double underlined text – example

`uwave`
: wavy underlined text – example

`dashuline`
: dashed underlined text – example

`dotuline`
: dotted underlined text – example

`sout`
: striked out text – ~~example~~ (default markup for deleted text)

`xout`
: crossed out text – *example*

`bf` bold text – **example**

`it` italic text – *example*

`sl` slanted text – *example*

`em` emphasized text – *example*

Call
`\usepackage[addedmarkup=`⟨*markup*⟩`]{changes}`
Examples
`\usepackage[addedmarkup=none]{changes}` ≙ `\usepackage{changes}`
`\usepackage[addedmarkup=uline]{changes}`

Call
`\usepackage[deletedmarkup=`⟨*markup*⟩`]{changes}`
Examples
`\usepackage[deletedmarkup=sout]{changes}` ≙ `\usepackage{changes}`
`\usepackage[deletedmarkup=xout]{changes}`
`\usepackage[deletedmarkup=uwave]{changes}`

4.1.5 authormarkup

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 are allowed:

`superscript`
: superscripted text – text$^{\text{author}}$ (default markup)

`subscript`
: subscripted text – text$_{\text{author}}$

`brackets`
: text in brackets – text(author)

`footnote`
: text in footnote – text[1]

`none`
: no author identification

[1] author

Call
  `\usepackage[authormarkup=⟨markup⟩]{changes}`
Examples
  `\usepackage[authormarkup=superscript]{changes}` ≙ `\usepackage{changes}`
  `\usepackage[authormarkup=subscript]{changes}`
  `\usepackage[authormarkup=brackets]{changes}`
  `\usepackage[authormarkup=footnote]{changes}`
  `\usepackage[authormarkup=none]{changes}`

4.1.6 authormarkupposition

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 are allowed:

`right`
: right of the text – text$^{\text{example}}$ (default value)

`left`
: left of the text – $^{\text{example}}$text

Call
  `\usepackage[authormarkupposition=⟨markup⟩]{changes}`
Examples
  `\usepackage[authormarkupposition=right]{changes}` ≙ `\usepackage{changes}`
  `\usepackage[authormarkupposition=left]{changes}`

4.1.7 authormarkuptext

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 are allowed:

`id`
: author's id – text$^{\text{id}}$ (default value)

`name`
: author's name – text$^{\text{authorname}}$

Call
  `\usepackage[authormarkuptext=⟨markup⟩]{changes}`
Examples
  `\usepackage[authormarkuptext=id]{changes}` ≙ `\usepackage{changes}`
  `\usepackage[authormarkuptext=name]{changes}`

4.1.8 ulem

All options for the *ulem* package can be specified as parameters of the `ulem`-option. Two or more options have to be put in curly brackets.

Call
```
\usepackage[ulem=⟨options⟩]{changes}
```
Examples
```
\usepackage[ulem=normalem]{changes}
\usepackage[ulem={normalem,normalbf}]{changes}
```

4.1.9 xcolor

All options for the *xcolor* package can be specified as parameters of the `xcolor`-option. Two or more option have to be embraced in curly brackets.

Call
```
\usepackage[xcolor=⟨options⟩]{changes}
```
Examples
```
\usepackage[xcolor=dvipdf]{changes}
\usepackage[xcolor={dvipdf,gray}]{changes}
```

4.2 Change management

4.2.1 \added

`\added`

The command `\added` marks new text. The new text is the mandatory argument for the command, thus it is written in curly braces. The optional argument contains key-value-pairs for author-id and remark. The author-id has to be defined using `\definechangesauthor`. If the remark contains special characters or spaces, use curly brackets to enclose the remark.

Call
```
\added[id=⟨author's id⟩, remark=⟨remark⟩]{⟨new text⟩}
```
Examples
```
This is \added[id=EK]{new} text.
```
This is new[EK] text.
```
This is \added[id=EK, remark={has to be in it}]{new} text.
```
This is new[EK](has to be in it) text.
```
This is \added[remark=anonymous]{new} text.
```
This is new(anonymous) text.

4.2.2 \deleted

\deleted

The command \deleted marks deleted text. For arguments see \added.

Call
\deleted[id=⟨*author's id*⟩, remark=⟨*remark*⟩]{⟨*deleted text*⟩}
Examples
```
This is \deleted[remark=obsolete]{bad} text.
```
This is ~~bad~~(obsolete) text.

4.2.3 \replaced

\replaced

The command \replaced marks replaced text. Mandatory arguments are the new text and the old text. For optional arguments see \added.

Call
\replaced[id=⟨*author's id*⟩, remark=⟨*remark*⟩]{⟨*new text*⟩}{⟨*old text*⟩}
Examples
```
This is \replaced[id=EK]{nice}{bad} text.
```
This is nice~~bad~~[EK] text.

4.2.4 \listofchanges

\listofchanges

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.

The style argument is optional, by default the list of changes is printed. If you want to print a summary you have to use the option style=summary.

new since

Call
```
\listofchanges[style=<list|summary>]
```

4.3 Author management

4.3.1 \definechangesauthor

\definechangesauthor

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

Call
```
\definechangesauthor[name={⟨author's name⟩}, color={⟨color⟩}]{⟨author's id⟩}
```
Examples
```
\definechangesauthor{EK}
\definechangesauthor[color=orange]{EK}
\definechangesauthor[name={Ekkart Kleinod}]{EK}
\definechangesauthor[name={Ekkart Kleinod}, color=orange]{EK}
```

4.4 Adaptation of the output

4.4.1 \setaddedmarkup

\setaddedmarkup

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".

Call
```
\setaddedmarkup{⟨definition⟩}
```
Examples
```
\setaddedmarkup{\emph{#1}}
\setaddedmarkup{+++:  #1}
```

4.4.2 \setdeletedmarkup

\setdeletedmarkup

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, deleted0 text can be used with "#1".

Call
  \setdeletedmarkup{⟨*definition*⟩}
Examples
  \setdeletedmarkup{\emph{#1}}
  \setdeletedmarkup{--: #1}

4.4.3 \setauthormarkup

\setauthormarkup

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's text can be used with "#1".

Call
  \setauthormarkup{⟨*definition*⟩}
Examples
  \setauthormarkup{(#1)}
  \setauthormarkup{(#1)~---~}
  \setauthormarkup{\marginpar{#1}}

4.4.4 \setauthormarkupposition

\setauthormarkupposition

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.

Possible values: *left* == left of the changes; all other values: right

Call
  \setauthormarkupposition{⟨*position*⟩}
Examples
  \setauthormarkupposition{left}

4.4.5 \setauthormarkuptext

\setauthormarkuptext

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

Possible values: *name* == author's name; all other values: author's id

Call
  \setauthormarkuptext{⟨*text*⟩}
Examples
  \setauthormarkuptext{name}

4.4.6 \setremarkmarkup

\setremarkmarkup

The command \setremarkmarkup defines the layout of remarks in the text. The default markup typesets the remark in a footnote.

Values for definition: any LaTeX-commands, author's id can be used with "#1", the remark can be shown using "#2". Using the author's id you can use the author's color with Changes@Color#1.

Call
  \setremarkmarkup{⟨*definition*⟩}
Examples
  \setremarkmarkup{(#2 -- #1)}
  \setremarkmarkup{\footnote{#1:\textcolor{Changes@Color#1}{#2}}}

4.4.7 \setsocextension

\setsocextension

e v2.0.0 The command \setsocextension sets the extension of the auxiliary file for the summary of changes (soc-file[2]). The default extension is "soc". In the example stated below, the soc-file for "foo.tex" would be named "foo.changes" instead of the default name "foo.soc".

Call
  \setsocextension{⟨*extension*⟩}
Examples
  \setsocextension{changes}

4.5 Other new commands

4.5.1 \textsubscript

\textsubscript

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

LaTeX provides the command `\textsuperscript`, but not it's counterpart `\textsubscript`. If the command is not defined yet, it will be provided by the *changes*-package. If the command is defined yet, it will not be changed.

Call
`\textsubscript{`⟨*text*⟩`}`
Examples
```
This is a \textsubscript{subscript} text.
```
This is a $_{\text{subscript}}$ text.

4.6 Used packages

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

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

xifthen
: provides an enhanced `\if`-command as well as a `while`-loop

xkeyval
: provides options with key-value-pairs

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

pdfcolmk
: loaded if colored text is used for markup (default markup); solves the problem of colored text and page breaks (with pdflatex)

ulem
: loaded if text has to be striked or exed out (default markup)

xcolor
: loaded if colored text is used for markup (default markup)

5 Authors

Several authors contributed to the *changes*-package. The authors are (in alphabetical order):

- Chiaradonna, Silvano
- Giovannini, Daniele
- Kleinod, Ekkart
- Mittelbach, Frank
- Wölfel, Philipp
- Wolter, Steve

6 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`

7 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/delcmdchanges.bash
```

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
```

8 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⟩
```

8.1 Package information and options

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

```
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{changes}
4 [2018/10/10 v2.1.0 changes package]
5 \typeout{*** changes package 2018/10/10 v2.1.0 ***}
```

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

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

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

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

8.1.1 Package options

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

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

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

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

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.

```
\newcommand{\Changes@optionaddedmarkup}{none}
\newcommand{\Changes@optiondeletedmarkup}{sout}
```

Option markup, sets markup options accordingly.

```
\newcommand{\Changes@optionmarkup}{default}
\DeclareOptionX{markup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{default}\or
\equal{#1}{underlined}\or
\equal{#1}{bfit}\or
\equal{#1}{nocolor}
}
{\renewcommand{\Changes@optionmarkup}{#1}}
{\PackageWarning{changes}{markup '#1' unknown, using '\Changes@optionmarkup'}}
}
\ifthenelse{\equal{\Changes@optionmarkup}{default}}
{
\renewcommand{\Changes@optionaddedmarkup}{none}
\renewcommand{\Changes@optiondeletedmarkup}{sout}
}
{}
\ifthenelse{\equal{\Changes@optionmarkup}{underlined}}
{
\renewcommand{\Changes@optionaddedmarkup}{uline}
\renewcommand{\Changes@optiondeletedmarkup}{sout}
}
{}
\ifthenelse{\equal{\Changes@optionmarkup}{bfit}}
{
\renewcommand{\Changes@optionaddedmarkup}{bf}
\renewcommand{\Changes@optiondeletedmarkup}{it}
}
{}
\ifthenelse{\equal{\Changes@optionmarkup}{nocolor}}
{
\renewcommand{\Changes@optionaddedmarkup}{uline}
\renewcommand{\Changes@optiondeletedmarkup}{sout}
}
{}
\typeout{changes-option 'markup=\Changes@optionmarkup'}
}
```

Option addedmarkup, stored or set to default value “none”.

```
\DeclareOptionX{addedmarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{none}\or
\equal{#1}{uline}\or
\equal{#1}{uuline}\or
\equal{#1}{uwave}\or
\equal{#1}{dashuline}\or
\equal{#1}{dotuline}\or
\equal{#1}{sout}\or
\equal{#1}{xout}\or
\equal{#1}{bf}\or
\equal{#1}{it}\or
\equal{#1}{sl}\or
\equal{#1}{em}
}
{\renewcommand{\Changes@optionaddedmarkup}{#1}}
{\PackageWarning{changes}{addedmarkup '#1' unknown, using '\Changes@optionaddedmarkup'}}
}
\typeout{changes-option 'addedmarkup=\Changes@optionaddedmarkup'}
}
```

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

```
\DeclareOptionX{deletedmarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{none}\or
\equal{#1}{uline}\or
\equal{#1}{uuline}\or
\equal{#1}{uwave}\or
\equal{#1}{dashuline}\or
\equal{#1}{dotuline}\or
\equal{#1}{sout}\or
\equal{#1}{xout}\or
\equal{#1}{bf}\or
\equal{#1}{it}\or
\equal{#1}{sl}\or
\equal{#1}{em}
}
{\renewcommand{\Changes@optiondeletedmarkup}{#1}}
{\PackageWarning{changes}{deletedmarkup '#1' unknown, using '\Changes@optiondeletedmarkup'}}
}
\typeout{changes-option 'deletedmarkup=\Changes@optiondeletedmarkup'}
```

```
}
```

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

```
\newcommand{\Changes@optionauthormarkup}{superscript}
\DeclareOptionX{authormarkup}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
\equal{#1}{superscript}\or
\equal{#1}{subscript}\or
\equal{#1}{brackets}\or
\equal{#1}{footnote}\or
\equal{#1}{none}
}
{\renewcommand{\Changes@optionauthormarkup}{#1}}
{\PackageWarning{changes}{authormarkup '#1' unknown, using '\Changes@optionauthormarkup'}
}
\typeout{changes-option 'authormarkup=\Changes@optionauthormarkup'}
}
```

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

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

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

```
\newcommand{\Changes@optionauthormarkuptext}{id}
\DeclareOptionX{authormarkuptext}{
\ifthenelse{\equal{\@empty}{#1}}
{}
{
\ifthenelse{
```

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

Options for package *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'}
}
```

8.1.2 Command options

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

`\definechangesauthor`

Declare available options of the command, define value storage.

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

Set the default values of the options.

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

\added

Declare available options of the command, define value storage.

```
\DeclareOptionX<Changes@added>{id}{\def\Changes@added@id{#1}}
\DeclareOptionX<Changes@added>{remark}{\def\Changes@added@remark{#1}}
\DeclareOptionX<Changes@added>{decision}{\def\Changes@added@dec{#1}}
\DeclareOptionX<Changes@added>{decisionid}{\def\Changes@added@decid{#1}}
\DeclareOptionX<Changes@added>{decisionremark}{\def\Changes@added@decrem{#1}}
```

Set the default values of the options.

```
\presetkeys{Changes@added}{
id=\@empty,
remark=\@empty,
decision=\@empty,
decisionid=\@empty,
decisionremark=\@empty
}{}
```

\deleted

Declare available options of the command, define value storage.

```
\DeclareOptionX<Changes@deleted>{id}{\def\Changes@deleted@id{#1}}
\DeclareOptionX<Changes@deleted>{remark}{\def\Changes@deleted@remark{#1}}
\DeclareOptionX<Changes@deleted>{decision}{\def\Changes@deleted@dec{#1}}
\DeclareOptionX<Changes@deleted>{decisionid}{\def\Changes@deleted@decid{#1}}
\DeclareOptionX<Changes@deleted>{decisionremark}{\def\Changes@deleted@decrem{#1}}
```

Set the default values of the options.

```
\presetkeys{Changes@deleted}{
id=\@empty,
remark=\@empty,
decision=\@empty,
decisionid=\@empty,
decisionremark=\@empty
}{}
```

\replaced

Declare available options of the command, define value storage.

```
\DeclareOptionX<Changes@replaced>{id}{\def\Changes@replaced@id{#1}}
\DeclareOptionX<Changes@replaced>{remark}{\def\Changes@replaced@remark{#1}}
\DeclareOptionX<Changes@replaced>{decision}{\def\Changes@replaced@dec{#1}}
\DeclareOptionX<Changes@replaced>{decisionid}{\def\Changes@replaced@decid{#1}}
\DeclareOptionX<Changes@replaced>{decisionremark}{\def\Changes@replaced@decrem{#1}}
```

Set the default values of the options.

```
\presetkeys{Changes@replaced}{
id=\@empty,
remark=\@empty,
decision=\@empty,
decisionid=\@empty,
decisionremark=\@empty
}{}
```

\listofchanges

Declare available options of the command, define value storage.

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

Set the default values of the options.

```
\presetkeys{Changes@loc}{
style=list
}{}
```

8.1.3 Option processing

Process the options.

```
\ProcessOptionsX*\relax
```

8.2 Packages

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

```
\newboolean{Changes@colored}
\setboolean{Changes@colored}{true}
\ifthenelse{\equal{\Changes@optionmarkup}{nocolor}}
{\setboolean{Changes@colored}{false}}
{}
\ifthenelse{\boolean{Changes@colored}}
{
\RequirePackage{xcolor}
\RequirePackage{pdfcolmk}
}
{}
```

Package *ulem* provides commands for striking out text.

```
\ifthenelse{
\equal{\Changes@optionaddedmarkup}{uline}\or
\equal{\Changes@optionaddedmarkup}{uuline}\or
\equal{\Changes@optionaddedmarkup}{uwave}\or
\equal{\Changes@optionaddedmarkup}{dashuline}\or
\equal{\Changes@optionaddedmarkup}{dotuline}\or
\equal{\Changes@optionaddedmarkup}{sout}\or
\equal{\Changes@optionaddedmarkup}{xout}\or
\equal{\Changes@optiondeletedmarkup}{uline}\or
\equal{\Changes@optiondeletedmarkup}{uuline}\or
\equal{\Changes@optiondeletedmarkup}{uwave}\or
\equal{\Changes@optiondeletedmarkup}{dashuline}\or
\equal{\Changes@optiondeletedmarkup}{dotuline}\or
\equal{\Changes@optiondeletedmarkup}{sout}\or
\equal{\Changes@optiondeletedmarkup}{xout}
}
{\RequirePackage[normalem,normalbf]{ulem}}
{}
```

8.3 Language dependent texts

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

```
\newcommand*\listofchangesname{List of changes}
\newcommand*\summaryofchangesname{Changes}
\newcommand*\changesaddname{Added}
\newcommand*\changesdeletename{Deleted}
\newcommand*\changesreplacename{Replaced}
\newcommand*\changesauthorname{Author}
\newcommand*\changesanonymousname{anonymous}
\newcommand*\changesnoloc{List of changes is available after the next \LaTeX\ run.}
\newcommand*\changesnosoc{Summary of changes is available after the next \LaTeX\ run.}
```

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

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

```
{
\addto\captionsngerman{\def\listofchangesname{Liste der \"Anderungen}}
\addto\captionsngerman{\def\summaryofchangesname{\"Anderungen}}
\addto\captionsngerman{\def\changesaddname{Eingef\"ugt}}
\addto\captionsngerman{\def\changesdeletename{Gel\"oscht}}
\addto\captionsngerman{\def\changesreplacename{Ersetzt}}
\addto\captionsngerman{\def\changesauthorname{Autor}}
\addto\captionsngerman{\def\changesanonymousname{Anonym}}
\addto\captionsngerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \LaTeX-Lauf verf\
\addto\captionsngerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \LaTeX-Lauf verf\"ugbar.}}

\addto\captionsgerman{\def\listofchangesname{Liste der \"Anderungen}}
\addto\captionsgerman{\def\summaryofchangesname{\"Anderungen}}
\addto\captionsgerman{\def\changesaddname{Eingef\"ugt}}
\addto\captionsgerman{\def\changesdeletename{Gel\"oscht}}
\addto\captionsgerman{\def\changesreplacename{Ersetzt}}
\addto\captionsgerman{\def\changesauthorname{Autor}}
\addto\captionsgerman{\def\changesanonymousname{Anonym}}
\addto\captionsgerman{\def\changesnoloc{Liste der \"Anderungen nach dem n\"achsten \LaTeX-Lauf verf\"
\addto\captionsgerman{\def\changesnosoc{\"Anderungen nach dem n\"achsten \LaTeX-Lauf verf\"ugbar.}}

\addto\captionsenglish{\def\listofchangesname{List of changes}}
\addto\captionsenglish{\def\summaryofchangesname{Changes}}
\addto\captionsenglish{\def\changesaddname{Added}}
\addto\captionsenglish{\def\changesdeletename{Deleted}}
\addto\captionsenglish{\def\changesreplacename{Replaced}}
\addto\captionsenglish{\def\changesauthorname{Author}}
\addto\captionsenglish{\def\changesanonymousname{anonymous}}
\addto\captionsenglish{\def\changesnoloc{List of changes is available after the next \LaTeX\ run.}}
\addto\captionsenglish{\def\changesnosoc{Summary of changes is available after the next \LaTeX\ run.}

\addto\captionsbritish{\def\listofchangesname{List of changes}}
\addto\captionsbritish{\def\summaryofchangesname{Changes}}
\addto\captionsbritish{\def\changesaddname{Added}}
\addto\captionsbritish{\def\changesdeletename{Deleted}}
\addto\captionsbritish{\def\changesreplacename{Replaced}}
\addto\captionsbritish{\def\changesauthorname{Author}}
\addto\captionsbritish{\def\changesanonymousname{anonymous}}
\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\changesaddname{Aggiunte}}
\addto\captionsitalian{\def\changesdeletename{Cancellazioni}}
\addto\captionsitalian{\def\changesreplacename{Sostituzioni}}
\addto\captionsitalian{\def\changesauthorname{Autore}}
\addto\captionsitalian{\def\changesanonymousname{anonimo}}
\addto\captionsitalian{\def\changesnoloc{La lista delle modifiche sar\`a disponibile alla prossima es
```

```
\addto\captionsitalian{\def\changesnosoc{Le modifiche sar\'a disponibile alla prossima es
}
{}
```

8.4 File extension

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

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

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

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

8.5 Authors

8.5.1 Author management

Author counter.

```
\newcounter{Changes@AuthorCount}
\setcounter{Changes@AuthorCount}{0}
\newcounter{Changes@Author}
```

\definechangesauthor Define a new author. Mandatory argument: author's id. Optional arguments (key-value): author's name (default: empty) and author's color (default: black).

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
\newcounter{Changes@AddCount#2}
\newcounter{Changes@DeleteCount#2}
\newcounter{Changes@ReplaceCount#2}
\ifthenelse{\boolean{Changes@colored}}
{
\expandafter
\let\csname Changes@AuthorColor#2\endcsname=\Changes@definechangesauthor@color
\colorlet{Changes@Color#2}{\@nameuse{Changes@AuthorColor#2}}
}
{}
}
```

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

```
\definechangesauthor[color=blue]{\@empty}
```

8.5.2 Author markup

kup@Author Store markup for authors.

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

thormarkup Set markup for authors.

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

tsubscript Define the command \textsubscript in case the author markup subscript is used and the command \textsubscript is not defined yet. \textsubscript is the antagonist of \textsuperscript which is predefined in LaTeX. The code is taken from LaTeX-FAQ 8.5.17.

```
\ifthenelse{\isundefined{\textsubscript}}
{
\DeclareRobustCommand*\textsubscript[1]{\@textsubscript{\selectfont#1}}
\newcommand{\@textsubscript}[1]{{\m@th\ensuremath{_{\mbox{\fontsize\sf@size\z@#1}}}}}
}{}
```

tauthormarkupposition Set position for author markup text.

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

\setauthormarkuptext Set author markup text to be displayed.

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

\Changes@Remark Markup of remarks. Default: in a footnote.

```
\newcommand{\Changes@Remark}[2]{%
\footnote{%
\ifthenelse{\not\equal{#1}{\@empty}}%
{#1: }%
{}%
#2%
}%
}
```

\setremarkmarkup Redefining the remark markup. Mandatory argument: markup definition.

```
\newcommand{\setremarkmarkup}[1]{%
\renewcommand{\Changes@Remark}[2]{#1}%
}
```

8.6 Change management commands

8.6.1 Text markup definition

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

\Changes@Markup@Added Store markup for added text.

```
\newcommand{\Changes@Markup@Added}[1]{%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{none}}{#1}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{uline}}{\uline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{uuline}}{\uuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{uwave}}{\uwave{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{dashuline}}{\dashuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{dotuline}}{\dotuline{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{sout}}{\sout{#1}}{}%
```

```
\ifthenelse{\equal{\Changes@optionaddedmarkup}{xout}}{\xout{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{bf}}{\textbf{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{it}}{\textit{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{sl}}{\textsl{#1}}{}%
\ifthenelse{\equal{\Changes@optionaddedmarkup}{em}}{\emph{#1}}{}%
}
```

ddedmarkup Set markup for added text.

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

up@Deleted Store markup for deleted text.

```
\newcommand{\Changes@Markup@Deleted}[1]{%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{none}}{#1}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{uline}}{\uline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{uuline}}{\uuline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{uwave}}{\uwave{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{dashuline}}{\dashuline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{dotuline}}{\dotuline{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{sout}}{\sout{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{xout}}{\xout{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{bf}}{\textbf{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{it}}{\textit{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{sl}}{\textsl{#1}}{}%
\ifthenelse{\equal{\Changes@optiondeletedmarkup}{em}}{\emph{#1}}{}%
}
```

etedmarkup Set markup for deleted text.

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

8.6.2 Change management command definition

ges@output Output command for the changed text. This command has the following arguments:

1. changed text (including markup)
2. unchanged text
3. author's id
4. remark
5. text for list of changes
6. change type for list of changes

7. decision (accept or reject)
8. decision author's id
9. decision remark

Define boolean for author test.

```
\newboolean{Changes@WrongID}
\newcommand{\Changes@output}[9]{%
```

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

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

Check if the author exists, error message otherwise. I have the feeling that this code is optimizable.

```
\setboolean{Changes@WrongID}{true}%
\setcounter{Changes@Author}{0}%
\whiledo{\value{Changes@Author} < \value{Changes@AuthorCount}}{%
\stepcounter{Changes@Author}%
\ifthenelse{\equal{#3}{\@nameuse{Changes@AuthorID\theChanges@Author}}}%
{\setboolean{Changes@WrongID}{false}}%
{}%
}%
\ifthenelse{\boolean{Changes@WrongID}}%
{\PackageError{changes}%
{Undefined changes author: #3}%
{You have to define the author #3 with e.g.: \definechangesauthor{#3}}}%
{}%
```

Check if the decision's author exists, error message otherwise.

```
\ifthenelse{\not\equal{#8}{\@empty}}%
{%
\setboolean{Changes@WrongID}{true}%
\setcounter{Changes@Author}{0}%
\whiledo{\value{Changes@Author} < \value{Changes@AuthorCount}}{%
\stepcounter{Changes@Author}%
\ifthenelse{\equal{#8}{\@nameuse{Changes@AuthorID\theChanges@Author}}}%
{\setboolean{Changes@WrongID}{false}}%
{}%
}%
\ifthenelse{\boolean{Changes@WrongID}}%
{\PackageError{changes}%
{Undefined changes author: #8}%
{You have to define the author #8 with e.g.: \definechangesauthor{#8}}}%
{}%
}%
{}%
```

Save author text for output.

```
\ifthenelse{\equal{\Changes@optionauthormarkuptext}{id}}%
{%
\@namedef{Changes@AuthorText}{#3}%
\ifthenelse{\not\equal{#8}{\@empty}}%
{\@namedef{Changes@DecAuthorText}{#8}}%
{}%
}%
{}%
\ifthenelse{\equal{\Changes@optionauthormarkuptext}{name}}%
{%
\@namedef{Changes@AuthorText}{\@nameuse{Changes@AuthorName#3}}%
\ifthenelse{\not\equal{#8}{\@empty}}%
{\@namedef{Changes@DecAuthorText}{\@nameuse{Changes@AuthorName#8}}}%
{}%
}%
{}%
```

Begin output

```
{%
```

Change color, if colored text is used.

```
\ifthenelse{\boolean{Changes@colored}}%
{\color{Changes@Color#3}}%
{}%
```

Output author text if author's id is given and text should appear left of changes.

```
\ifthenelse{\equal{\Changes@optionauthormarkupposition}{left} \and \not\equal{#3}{\@empty}}%
{\Changes@Markup@Author{\@nameuse{Changes@AuthorText}}}%
{}%
```

Output changed text.

```
{#1}%
```

Output author text if author's id is given and text should appear right of changes.

```
\ifthenelse{\equal{\Changes@optionauthormarkupposition}{right} \and \not\equal{#3}{\@empty}}%
{\Changes@Markup@Author{\@nameuse{Changes@AuthorText}}}%
{}%
```

Output remark if a remark is given.

```
\ifthenelse{\not\equal{#4}{\@empty}}%
{\Changes@Remark{#3}{#4}}%
{}%
}%
```

Store line for list of changes.

```
\ifthenelse{\equal{\@empty}{#3}}%
{\def\Changes@locid{}}%
{\def\Changes@locid{~(#3)}}%
\addcontentsline{loc}{subsection}{#6\Changes@locid: \truncate{.3\textwidth}{#5}}%
}%
```

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

```
{#2}%
}
```

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

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

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

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

```
\setkeys{Changes@added}{#1}%
\let\Changes@esphack\relax%
\Changes@output%
{\Changes@Markup@Added{#2}}%
{%
\ifthenelse{\equal{\@empty}{#2}}%
{\@bsphack \let\Changes@esphack\@esphack}%
{#2}%
}%
{\Changes@added@id}%
{\Changes@added@remark}%
{#2}%
{\changesaddname}%
{\Changes@added@dec}%
{\Changes@added@decid}%
{\Changes@added@decremark}%
\stepcounter{Changes@AddCount\Changes@added@id}%
\Changes@esphack%
}
```

`\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, remark, decision, decision author's id, decision remark

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

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

```
\setkeys{Changes@deleted}{#1}%
\let\Changes@esphack\relax%
\Changes@output%
{\Changes@Markup@Deleted{#2}}%
{\@bsphack \let\Changes@esphack\@esphack}%
{\Changes@deleted@id}%
{\Changes@deleted@remark}%
{#2}%
{\changesdeletename}%
{\Changes@deleted@dec}%
{\Changes@deleted@decid}%
{\Changes@deleted@decremark}%
\stepcounter{Changes@DeleteCount\Changes@deleted@id}%
\Changes@esphack%
}
```

\replaced The command formats text as replaced text.

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

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

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

```
\setkeys{Changes@replaced}{#1}%
\let\Changes@esphack\relax%
\Changes@output%
{{\Changes@Markup@Added{#2}}{\Changes@Markup@Deleted{#3}}}%
{%
\ifthenelse{\equal{\@empty}{#2}}%
{\@bsphack \let\Changes@esphack\@esphack}%
{#2}%
}%
{\Changes@replaced@id}%
{\Changes@replaced@remark}%
{#2}%
{\changesreplacename}%
{\Changes@replaced@dec}%
{\Changes@replaced@decid}%
{\Changes@replaced@decremark}%
\stepcounter{Changes@ReplaceCount\Changes@replaced@id}%
\Changes@esphack%
```

```
526 }
```

8.7 List of changes

The list of changes truncates text, therefore the *truncate* package is used.

```
527 \RequirePackage[breakall]{truncate}
```

`\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. The contents have to be separated by a semicolon.

```
528 \def\changes@chopline#1;#2;#3;#4;#5;#6 \\{
529 \def\Changes@InID{#1}
530 \def\Changes@InColor{#2}
531 \def\Changes@InName{#3}
532 \def\Changes@InAdded{#4}
533 \def\Changes@InDeleted{#5}
534 \def\Changes@InReplaced{#6}
535 }
```

`\listofchanges` This command outputs the list of changes.

Two styles are available: `list` (default) and `summary`. `list` prints the list of changes lika a list of figures. `summary` prints a summary of changes separated by authors.

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

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

Some definitions that have to reside outside the command in order to use the command multiple times. In further versions: compute length of bounding box für summary.

```
536 \newlength{\Changes@Len@summ}
537 \setlength{\Changes@Len@summ}{.2\textwidth}
538 \newboolean{Changes@MoreLines}
```

The definition of `\listofchanges`.

```
539 \newcommand{\listofchanges}[1][style=list]{%
540 \setkeys{Changes@loc}{#1}%
```

All work is done only in draft mode.

```
541 \ifthenelse{\boolean{Changes@optiondraft}}%
542 {%
```

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

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

Print list.

```
\ifthenelse{\equal{\Changes@loc@style}{list}}%
{%
\section*{\listofchangesname}%
\IfFileExists{\jobname.loc}%
{}{%
\emph{\changesnoloc}%
\PackageWarning{changes}{LaTeX rerun needed for list of changes}%
}%
\@starttoc{loc}{}%
}%
```

Print summary, but only in draft mode.

```
{%
\section*{\summaryofchangesname}%
\IfFileExists{\jobname.\Changes@extension}%
{%
\setboolean{Changes@MoreLines}{true}%
\newread\Changes@InFile%
\openin\Changes@InFile = \jobname.\Changes@extension%
\whiledo{\boolean{Changes@MoreLines}}{%
\read\Changes@InFile to \Changes@Line%
\ifeof\Changes@InFile%
\setboolean{Changes@MoreLines}{false}%
\else%
\expandafter\changes@chopline\Changes@Line\\%
\textbf{%
\ifthenelse{\boolean{Changes@colored}}%
{\color{\Changes@InColor}}%
{}%
\ifthenelse{\equal{\Changes@InID}{\@empty}}%
{\changesauthorname: \changesanonymousname}%
```

```
{%
\changesauthorname: \Changes@InID%
\ifthenelse{\equal{\Changes@InName}{\@empty}}%
{}%
{ (\Changes@InName)}%
}%
}\\%
\parbox{\Changes@Len@summ}{\changesaddname~\dotfill~\Changes@InAdded}\\%
\parbox{\Changes@Len@summ}{\changesdeletename~\dotfill~\Changes@InDeleted}\\%
\parbox{\Changes@Len@summ}{\changesreplacename~\dotfill~\Changes@InReplaced}\\[1ex]%
\fi%
}%
\closein\Changes@InFile%
}{%
\emph{\changesnosoc}%
\PackageWarning{changes}{LaTeX rerun needed for summary of changes}%
}%
}%
```

In final mode print nothing.

```
}{}%
}
```

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

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

```
\AtEndDocument{
```

Open output file.

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

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

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

Output data for list of changes.

```
\setcounter{Changes@Author}{0}
\whiledo{\value{Changes@Author} < \value{Changes@AuthorCount}}{%
\stepcounter{Changes@Author}%
\def\Changes@ID{\@nameuse{Changes@AuthorID\theChanges@Author}}%
```

```
\immediate\write\Changes@OutFile{\Changes@ID;%
\@nameuse{Changes@AuthorColor\Changes@ID};%
\@nameuse{Changes@AuthorName\Changes@ID};%
\the\value{Changes@AddCount\Changes@ID};%
\the\value{Changes@DeleteCount\Changes@ID};%
\the\value{Changes@ReplaceCount\Changes@ID}}%
}%
```

Close output file.

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

Restore original definition of `\protect`.

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

⟨/changes⟩