THE CNLTX BUNDLE

Documentation for LaTeX 2ε Packages or Classes

v0.15 2019/11/01

LaTeX tools and documenting facilities the CN way

Clemens Niederberger

https://github.com/cgnieder/cnltx

contact@mychemistry.eu

A versatile bundle of packages and classes for consistent formatting of control sequences, package options, source code examples, and writing a package manual (including an index containing the explained control sequences, options, …).

The bundle also provides several other small ideas of mine such as a mechansim for providing abbreviations *etc.* Not at least it provides a number of programming tools.

Table of Contents

Part I.
About The Bundle

1. Background

The CNLTX bundle contains different packages and classes.[1] I developed it as a successor of my class cnpkgdoc [Nie13] that I used until now for writing the documentation of my packages. The intention behind the new bundle is a cleaner interface and less unnecessary ballast, hence the separation into packages and classes. This is actually a bit of a contradiction: the document class CNLTX-DOC loads *all* packages of the bundle which makes it more feature-rich than cnpkgdoc ever used to be. The bundle provides source code environments that also print the output and defines quite a lot of macros for formatting of control sequence names, package names, package options and so on.

Part of the motivation is also that users have asked me how I created the manuals for my packages. Now I can refer to this bundle.

Another reason for the splitting into separate packages is – besides the advantage of easier maintenance – is that I wanted to add programming tools that I often use into CNLTX-BASE which may allow me (and others) to use them for other packages, too, without having to define them each time. So it is quite likely that CNLTX-BASE will get extended in the future.

The bundle provides listings style for LaTeX code, bibliography database files and index style files. It provides a biblatex citation and bibliography style closely linked to CNLTX-DOC. It provides a bibliography database file containing many LaTeX packages. It provides… Let's stop here. You see that the bundle provides a lot of different features which explains why this manual is more than 60 pages long.

The most detailed documentation for the bundle is as always the source code of the `sty` and `cls` files but I'm trying to provide a documentation as comprehensive as possible. Reading the source files may show how things are implemented but the intended use only becomes clear when you read this manual.

The bundle reflects the fact that I haven't started using literate programming, yet. I don't use `docstrip` and don't write `dtx` files but always write the `sty` or `cls` files directly. I write the manual always at the same time but as a separate file. While I'm entirely aware of the advantages of literate programming I never could bring myself to start to use it myself. As a consequence I have no idea if this bundle can be used for it or not.

Source code formatting is done with the help of the powerful listings package [HM19] by Carsten Heinz and later Brooks Moses, now maintained by Jobst Hoffmann. The only real drawback I have found with it is recognizing starred und un-starred versions of an environment as different keywords. This does not seem to be possible which is why indexing of such environments will lead to wrong page numbers.

The fancy frames of the source code examples are realized with the mdframed package by Marco Daniel [DS13], loaded with the option `frametheme = tikz`.

1. Well, *one* class for the time being,

Besides all this I included some other ideas of mine in this bundle which are all provided by CNLTX-TOOLS. This includes a mechansim for defining clever abbreviations or macros that make it easy to index names the same way biblatex does.

2. Bundled Packages, Classes and Files

The CNLTX bundle currently bundles the following packages, classes and files:

Introduced in version 0.9

- CNLTX – a wrapper package for usage in documents. It loads one or more of the following packages. See section 4 for details on the usage.
  `\usepackage{cnltx}`

- CNLTX-BASE – a package that defines base macros for error-messaging, expansion control, tokenlist manipulation and defining of expandable macros. It also provides color definitions and defines a few color schemes for the CNLTX-DOC class. All other packages and classes of the CNLTX bundle load this package. This package can be used stand-alone.
  `\usepackage{cnltx-base}`
  The packages commands are not described in the main part of this documentation but only in section A.1, *i. e.*, in the appendix.

- CNLTX-DOC – a class for writing package manuals. Loads CNLTX-EXAMPLE and CNLTX-TOOLS and implicitly all other files of the bundle.
  `\documentclass{cnltx-doc}`

- CNLTX-EXAMPLE – a package that defines macros and environments for describing control sequences and options and for including source code. Loads CNLTX-LISTINGS. This package can be used stand-alone.
  `\usepackage{cnltx-example}`

Introduced in version 0.4

- CNLTX-LISTINGS – a package that defines the listings language 'BibTeX'. Also defines a list of highlighted control sequence names and environment names, loaded by CNLTX-EXAMPLE. The additional control sequence and environment names used to be defined in CNLTX-CSNAMES. That package got removed and its contents are now provided by CNLTX-LISTINGS. This package can be used stand-alone.
  `\usepackage{cnltx-listings}`

Introduced in version 0.2

- CNLTX-TOOLS – a package that defines tools used by CNLTX-DOC that are unrelated to LaTeX documentation *per se*. This package can be used stand-alone.
  `\usepackage{cnltx-tools}`

Introduced in version 0.11

- CNLTX-TRANSLATIONS – a package that provides translations needed by the other modules. It makes no sense to use this package standalone although it's possible.

- `cnltx.ist` – an index style file that is used when the option `add-index` for CNLTX-DOC is activated and the option `index-style` is not used.

Introduced in version 0.4

- `cnltx.bib` – a bibliography file that contains a small but growing number of package entries, see section D. Used by CNLTX-DOC when the `add-bib` is used.

Introduced in version 0.4

- `cnltx.bbx`, `cnltx.cbx` and `cnltx.dbx` – files related to the biblatex style `cnltx`. The biblatex style defined in those files is used when the `add-bib` for CNLTX-DOC is used.

3. License and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

The CNLTX-BASE package loads the following packages: pgfopts[2] [Wri14], etoolbox[3] [Leh19b], ltxcmds[4] [Obe16b], pdftexcmds[4] [Obe19], trimspaces[5] [Rob09] and xcolor[6] [Ker16].

The CNLTX-DOC class loads the packages CNLTX-BASE, CNLTX-EXAMPLE, CNLTX-TRANSLATIONS, ulem[7] [Ars11], multicol[8] [Mit19], ragged2e[9] [Sch19b], marginnote[10] [Koh18] and hyperref[11] [OR19]. It is a wrapper class for the KOMA-Script class scrartcl[12] [Koh19]. The class has the option `load-preamble` which when used will load additional packages, see section 10.5 on page 41 for details.

The CNLTX-EXAMPLE package loads the packages: CNLTX-BASE, CNLTX-LISTINGS, CNLTX-TOOLS, CNLTX-TRANSLATIONS, mdframed[13] [DS13], textcomp[14] [Rah16], idx-cmds[15] [Nie15], ifxetex[16] [Rob10], adjustbox[17] [Sch19a].

The CNLTX-LISTINGS package loads the packages CNLTX-BASE, listings[18] [HM19] and catchfile[19] [Obe16a].

The CNLTX-TOOLS package loads the packages CNLTX-BASE, CNLTX-TRANSLATIONSand accsupp[4] [Obe18].

CNLTX-TRANSLATIONS loads the translations package [Nie17].

All other packages that are loaded are loaded by the mentioned packages and not directly by any of the packages or classes of the CNLTX bundle. Like all of my packages CNLTX implicitly relies on an up to date TeX distribution.

2. on CTAN as `pgfopts`: `http://mirrors.ctan.org/macros/latex/contrib/pgfopts/`
3. on CTAN as `etoolbox`: `http://mirrors.ctan.org/macros/latex/contrib/etoolbox/`
4. on CTAN as `oberdiek`: `http://mirrors.ctan.org/macros/latex/contrib/oberdiek/`
5. on CTAN as `trimspaces`: `http://mirrors.ctan.org/macros/latex/contrib/trimspaces/`
6. on CTAN as `xcolor`: `http://mirrors.ctan.org/macros/latex/contrib/xcolor/`
7. on CTAN as `ulem`: `http://mirrors.ctan.org/macros/latex/contrib/ulem/`
8. on CTAN as `multicol`: `http://mirrors.ctan.org/macros/latex/required/tools/multicol/`
9. on CTAN as `ragged2e`: `http://mirrors.ctan.org/macros/latex/contrib/ms/ragged2e/`
10. on CTAN as `marginnote`: `http://mirrors.ctan.org/macros/latex/contrib/marginnote/`
11. on CTAN as `hyperref`: `http://mirrors.ctan.org/macros/latex/contrib/hyperref/`
12. on CTAN as `koma-script`: `http://mirrors.ctan.org/macros/latex/contrib/koma-script/`
13. on CTAN as `mdframed`: `http://mirrors.ctan.org/macros/latex/contrib/mdframed/`
14. on CTAN as `textcomp`: `http://mirrors.ctan.org/macros/latex/contrib/textcomp/`
15. on CTAN as `idxcmds`: `http://mirrors.ctan.org/macros/latex/contrib/idxcmds/`
16. on CTAN as `ifxetex`: `http://mirrors.ctan.org/macros/latex/contrib/ifxetex/`
17. on CTAN as `adjustbox`: `http://mirrors.ctan.org/macros/latex/contrib/adjustbox/`
18. on CTAN as `listings`: `http://mirrors.ctan.org/macros/latex/contrib/listings/`
19. on CTAN as `catchfile`: `http://mirrors.ctan.org/macros/latex/contrib/catchfile/`

4. Usage of the Bundle

The intended use of this bundle is three-fold:

- The main use-case is documenting my own LaTeX packages. This is done with

```
\documentclass{cnltx-doc}
```

  and actually loads most if not all of the bundle.

- The module CNLTX-BASE is also intended as a programming tools package that will be used in other packages eventually. For example it is used by the cntformats package.

- In case parts of this bundle prove useful to be used in a document the recommended way is to add

```
\usepackage{cnltx}
```

  to the preamble which will load the CNLTX-BASE module. Other needed modules can be given as package option by using the name part after the dash as option.

```
\usepackage[example]{cnltx}
```

  would load CNLTX-EXAMPLE.

- Parts of the bundle – especially CNLTX-BASE – may prove useful in other packages. The loading the packages directly as indicated in section 3 seems the best way. After loading CNLTX-BASE the other modules can also be loaded with `\cnltx@load@module`, see section A.1.1 for details.

Part II.
Details of Available Commands, Environments and Options

5. Options and Setup

The CNLTX bundle has a large number of options. The CNLTX-DOC class only knows a few options (described in section 10.1 on page 38) as *class* options, though. All other options regardless if they're defined by a package or a class can and should be set with the setup command:

`\setcnltx{`⟨*options*⟩`}`
Setup command for the CNLTX bundle. This command is provided by CNLTX-BASE.

The source code environments defined by the CNLTX-EXAMPLE package also have optional arguments that can be used to set the options for the environment locally.

6. Available Commands

6.1. Description of Macros, Environments and Options

provided by CNLTX-EXAMPLE

The commands described in this section all are provided by the CNLTX-EXAMPLE package. They all are related to the typesetting of provided macros, options and the like.

`\code{`⟨*arg*⟩`}`
Formatting of source code. This is *no* verbatim command. Used internally in the following commands.

`\verbcode`⟨*char*⟩⟨*code*⟩⟨*char*⟩

Introduced in version 0.2

A verbatim command that uses the same formatting as the source code example environments, *cf.* section 8.4. This is a wrapper for `\lstinline` which loads the corresponding style.

`\cs*{`⟨*name*⟩`}`
Format the control sequence ⟨*name*⟩, `\cs{name}`: `\name`. Adds a corresponding index entry. The starred form does not add an index entry.

`\csidx{`⟨*name*⟩`}`
Adds an index entry but does not typeset the control sequence ⟨*name*⟩.

`\env*{`⟨*name*⟩`}`
Format the environment ⟨*name*⟩, `\env{name}`: `name`. Adds a corresponding index entry with a hint that the entry refers to an environment. The starred form does not add an index entry.

`\envidx{`⟨*name*⟩`}`
Adds an index entry but does not typeset the environment ⟨*name*⟩.

`\meta{`⟨*meta*⟩`}`
Description of an argument, `\meta{meta}`: ⟨*meta*⟩.

`\marg{`⟨*arg*⟩`}`
A mandatory argument. ⟨*arg*⟩ is formatted with `\meta` if it is not blank, `\marg{arg}`: {⟨*arg*⟩}.

`\Marg{`⟨*arg*⟩`}`
Introduced in version 0.2
A mandatory argument. ⟨*arg*⟩ is formatted with `\code` if it is not blank, `\Marg{arg}`: {arg}.

`\oarg{`⟨*arg*⟩`}`
An optional argument. ⟨*arg*⟩ is formatted with `\meta` if it is not blank, `\oarg{arg}`: [⟨*arg*⟩].

`\Oarg{`⟨*arg*⟩`}`
Introduced in version 0.2
An optional argument. ⟨*arg*⟩ is formatted with `\code` if it is not blank, `\Oarg{arg}`: [arg].

`\darg{`⟨*arg*⟩`}`
An argument with parentheses as delimiters. ⟨*arg*⟩ is formatted with `\meta` if it is not blank, `\darg{arg}`: (⟨*arg*⟩).

`\Darg{`⟨*arg*⟩`}`
Introduced in version 0.2
An argument with parentheses as delimiters. ⟨*arg*⟩ is formatted with `\code` if it is not blank, `\Darg{arg}`: (arg).

`\sarg`
An optional star argument, `\sarg`: ∗.

`\newarg[`⟨*arg formatting*⟩`]{`⟨*cs*⟩`}{`⟨*left delim*⟩`}{`⟨*right delim*⟩`}` Default: `\meta`
Changed in version 0.2
Command used to define the argument commands: `\newarg\marg{\{}{\}}`. The optional argument determines how the argument of the new command will be formatted. This is done with `\meta` per default. `\Marg` is defined `\newarg[\code]\Marg{\{}{\}}`.

`\option∗{`⟨*name*⟩`}`
An option ⟨*name*⟩, `\option{name}`: name. Adds a corresponding index entry. The starred form does not add an index entry.

`\optionidx{`⟨*name*⟩`}`
Adds an index entry but does not typeset the option ⟨*name*⟩.

`\module∗{`⟨*name*⟩`}`
A module ⟨*name*⟩, `\module{name}`: name. Adds a corresponding index entry. The starred form does not add an index entry. In some of my packages I like to organize options by grouping them in different classes that I call “modules”. This command refers to those modules.

`\moduleidx∗{`⟨*name*⟩`}`
Adds an index entry but does not typeset the option ⟨*name*⟩.

`\key*-{`⟨*name*⟩`}{`⟨*value*⟩`}`
A key ⟨*name*⟩ with value ⟨*value*⟩, the optional star prevents an index entry, the optional - strips the braces around ⟨*value*⟩; `\key{key}{value}`: `key = {`⟨*value*⟩`}`; `\key-{key}{value}`: `key =` ⟨*value*⟩

`\keyis*-{`⟨*name*⟩`}{`⟨*value*⟩`}`
Introduced in version 0.2
A key ⟨*name*⟩ set to value ⟨*value*⟩, the optional star prevents an index entry, the optional - strips the braces around `value`; `\key{keyis}{value}`: `key = {value}`.

`\choices{`⟨*clist of choices*⟩`}`
A list of choices, `\choices{one,two,three}`: `one|two|three`

`\choicekey{`⟨*name*⟩`}{`⟨*clist of choices*⟩`}`
A key ⟨*name*⟩ with a list of possible values, `\choicekey{key}{one,two,three}`: `key = one|two|three`

`\boolkey{`⟨*name*⟩`}`
A boolean key ⟨*name*⟩ with choices `true` and `false`, `\boolkey{key}`: `key = true|false`

`\default{`⟨*value*⟩`}`
Markup for a default choice, `\choices{one,\default{two},three}`: `one|two|three`

6.2. Versioning Commands, Licensing and Related Stuff

provided by CNLTX-DOC
The commands described in this section are provided by the CNLTX class except where indicated differently. These commands are related to information about the legal stuff of a package and where to find it on th world wide web.

`\sinceversion{`⟨*version*⟩`}`
Introduced in version 0.0
Gives a sidenote like the one on the left.

`\changedversion{`⟨*version*⟩`}`
Changed in version 0.0
Gives a sidenote like the one on the left.

`\newnote*{`⟨*cs*⟩`}[`⟨*num*⟩`][`⟨*optional*⟩`]{`⟨*definition*⟩`}`
Defines a note like `\sinceversion`. The syntax of the command is the same as the one of `\newcommand`. `\sinceversion` was defined as follows:
`\newnote*\sinceversion[1]{Introduced in version~#1}`
or actually like this:
`\newnote*\sinceversion[1]{\GetTranslation{cnltx-introduced}~#1}`

`\newpackagename{`⟨*cs*⟩`}{`⟨*name*⟩`}`
Define a comand ⟨*cs*⟩ that prints ⟨*name*⟩ formatted like CNLTX, *i. e.* in small caps and colored with the color `cnltx` (see section 13.2).

`\lppl`
Typesets "LPPL" and adds a corresponding index entry.

`\LPPL`
Typesets "LaTeX Project Public License" and adds the same index entry as `\lppl`.

`\license*[`⟨*maintenance status*⟩`]` Default: `maintained`
Changed in version 0.2
Typesets 'Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."'. The un-starred variant adds a `\par`.

`\ctan`
Typesets "CTAN" and adds a corresponding index entry.

`\CTAN`
Typesets "Comprehensive TeX Archive Network" and adds the same index entry as `\ctan`.

`\pkg*{`⟨*package*⟩`}`
provided by CNLTX-EXAMPLE
Format the package name ⟨*package*⟩ and add an index entry. The starred variant adds nothing to the index.

`\pkgidx{`⟨*package*⟩`}`
provided by CNLTX-EXAMPLE
Add an index entry for the package ⟨*package*⟩.

`\cls*{`⟨*class*⟩`}`
provided by CNLTX-EXAMPLE
Format the class name ⟨*class*⟩ and add an index entry. The starred variant adds nothing to the index.

`\clsidx{`⟨*class*⟩`}`
provided by CNLTX-EXAMPLE
Add an index entry for the class ⟨*class*⟩.

`\CTANurl[`⟨*directory*⟩`]{`⟨*name*⟩`}`
Writes a CTAN link like the ones in section 3 on page 5 in the footnotes. The predefined directory is `macros/latex/contrib`. The link address will be:

`http://mirrors.ctan.org/`⟨*directory*⟩`/`⟨*name*⟩`/`.

`\email{`⟨*email address*⟩`}`
Introduced in version 0.11
A wrapper for `\href{mailto:#1}{#1}`.

`\website{`⟨*web address*⟩`}`
Introduced in version 0.11
A wrapper for `\href{http://#1/}{#1}`.

`\securewebsite{`⟨*web address*⟩`}`
Introduced in version 0.11
A wrapper for `\href{https://#1/}{#1}`.

`\needpackage[`⟨*directory*⟩`]{`⟨*name*⟩`}`
Introduced in version 0.2
A wrapper for `\pkg{#2}\footnote{\CTANurl[#1]{#2}}`

`\needclass[`⟨*directory*⟩`]{`⟨*name*⟩`}`
Introduced in version 0.2
A wrapper for `\cls{#2}\footnote{\CTANurl[#1]{#2}}`

```
\newpackagename{\foothree}{foo-3}%
now \foothree\ looks like \cnltx.
```

now FOO-3 looks like CNLTX.

6.3. Input Source Code Files

Similar to the environments described in section 7.2 on the next page CNLTX-EXAMPLE provides a few commands for inputting source code files, formatting and printing the source code and inputting the file directly.

`\inputexample[`⟨*options*⟩`]{`⟨*file name*⟩`}`
The equivalent of the `example` environment, see section 7.2 on the following page.

`\inputsidebyside[`⟨*options*⟩`]{`⟨*file name*⟩`}`
The equivalent of the `sidebyside` environment, see section 7.2 on the next page.

`\inputsourcecode[`⟨*options*⟩`]{`⟨*file name*⟩`}`
The equivalent of the `sourcecode` environment, see section 7.2 on the following page.

`\implementation[`⟨*options*⟩`]{`⟨*file name*⟩`}`
Introduced in version 0.5
A wrapper for `\lstinputlisting[style=cnltx,#1]{#2}`

It is possible to define further commands like this:

`\newinputsourcefilecmd[`⟨*option*⟩`]{`⟨*control sequence*⟩`}`
Defines ⟨*control sequence*⟩ as a new source code input command where ⟨*options*⟩ are preset.

The existing commands have been defined like this:

```
\newinputsourcefilecmd\inputexample
\newinputsourcefilecmd[side-by-side]\inputsidebyside
\newinputsourcefilecmd[code-only]\inputsourcecode
```

7. Available Environments

7.1. Description Environments

CNLTX-DOC defines some description environments used to describe macros, environments or options.

`\begin{commands}`
A description-like environment for describing commands. While this environment is a list internally and thus recognizes `\item` own commands are used to describe macros. They are explained in section 8.1 on the next page.

`\begin{options}`
A description-like environment for describing options. While this environment is a list internally and thus recognizes `\item` own commands are used to describe options. They are explained in section 8.2 on page 15.

`\begin{environments}`
A description-like environment for describing environments. While this environment is a list internally and thus recognizes `\item` own commands are used to describe environments. They are explained in section 8.3 on page 17.

These environments are lists all using the same internal `\list`. The setup of this list can be changed via an option:

`list-setup` = {⟨*definitions*⟩}
Default: `\leftmargin=0pt \labelwidth=2em \labelsep=0pt \itemindent=-1em`
The setup of the `\list` used by the `commands`, `options` and `environments` environments.

7.2. Source Code Environments

CNLTX-EXAMPLE defines the following environments that are used to display source code and possibly the output of the source code, too.

`\begin{example}[`⟨*options*⟩`]`
This environment is a formatted verbatim environment that also inputs the output of the inputted code. This environment is described in section 8.4 on page 18.

`\begin{sidebyside}[`⟨*options*⟩`]`
This environment is a formatted verbatim environment that also inputs the output of the inputted code. Source and output are printed side-by-side. This environment is described in section 8.4 on page 18.

`\begin{sourcecode}[`⟨*options*⟩`]`
This environment is a formatted verbatim environment. This environment is described in section 8.4 on page 18.

Introduced in version 0.2

In each of these environments certain hooks are provided that can be used to add definitions you like:

`pre-code` = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed before the source code is inserted.

`after-code` = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed after the source code is inserted.

`pre-output` = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed before the output of the source code is inserted.

`after-output` = {⟨*definitions*⟩}
⟨*definitions*⟩ are placed after the output of the source code is inserted.

It is possible to define further environments like this:

`\newsourcecodeenv[`⟨*option*⟩`]{`⟨*name*⟩`}`
Defines ⟨*name*⟩ as a new source code environment where ⟨*options*⟩ are preset.

The existing environments have been defined like this:

```
\newsourcecodeenv{example}
\newsourcecodeenv[side-by-side]{sidebyside}
\newsourcecodeenv[code-only]{sourcecode}
```

8. Usage of the Various Functions

8.1. Command Descriptions

Inside of the environment `commands` that was introduced in section 7.1 on page 11 items are input via the following command:

`\command*{`⟨*name*⟩`}[`⟨*stuff after*⟩`]`
This macro formats a control sequence with `\cs` and puts a line break after it. The optional argument allows printing things directly after the command name and can thus be used for adding arguments. The star prevents the creation of an index entry.

`\Default*!{`⟨*code*⟩`}`
Changed in version 0.3
This command can be placed after `\command` or `\opt` in order to give a default definition of a macro or a default value of an option. The definition will then be placed on the same line flush right. The star prevents the insertion of `\newline` after it. The optional bang adds the information that an option is mandatory, *i. e.* has to be set.

`\expandable`
Introduced in version 0.5
Adds the symbol * to the left of a command in the margin to indicate that the command is expandable. This command should be used *immediately* before `\command`.

`\unexpandable`
Introduced in version 0.5
Adds the symbol * to the left of a command in the margin to indicate that the command is not expandable. This command should be used *immediately* before `\command`.

`\expandablesign` Default: `\textasteriskcentered`
Introduced in version 0.5
The macro that holds the sign used by `\expandable` and `\unexpandable`.

`\expandablesymbol`

Introduced in version 0.11

The symbol *, *i. e.*, `\expandablesign` formatted with the color `expandable`.

`\unexpandablesymbol`

Introduced in version 0.11

The symbol *, *i. e.*, `\expandablesign` formatted with the color `unexpandable`.

```
\begin{commands}
  \command{cs}
    This is about foo bar baz.
  \command{cs}[\marg{arg}]
    This one has an argument.
  \command{cs}[\sarg\oarg{option}]
    This has a star variant and an optional argument.
  \command{cs}\Default{foo bar}
    This one has the default replacement text \code{foo bar}
  \expandable\command{cs}
    This macro is expandable.
\end{commands}
```

`\cs`

This is about foo bar baz.

`\cs{`⟨*arg*⟩`}`

This one has an argument.

`\cs*[`⟨*option*⟩`]`

This has a star variant and an optional argument.

`\cs` Default: `foo bar`

This one has the default replacement text `foo bar`

* `\cs`

This macro is expandable.

The `\expandablesign` can of course be redefined to something else you like better. For the sake of completeness there is an option that does exactly this:

`expandable-sign` = {⟨*definition*⟩} Default: `\textasteriskcentered`

Introduced in version 0.5

Redefines `\expandablesign` to ⟨*definition*⟩.

8.2. Option Descriptions

The `options` environment knows a few more commands to meet all the different kinds of options.

`\opt*`
An option. The star prevents an index entry.

`\keyval*-{`⟨*key*⟩`}{`⟨*value*⟩`}`
A key/value option. The optional star prevents an index entry. The optional `-` strips the braces around ⟨*value*⟩, see the example below.

`\keychoice*{`⟨*key*⟩`}{`⟨*list of choices*⟩`}`
A key/value option where the value is one of a list of choices. The star prevents an index entry.

`\keybool*{`⟨*name*⟩`}`
A boolean key, that ist a choice key with choices `true` and `false`. The star prevents an index entry.

`\Default*!{`⟨*code*⟩`}`
Changed in version 0.3
This command can be placed after `\command` or `\opt` (or any of the other commands for adding an option to the `options` list) in order to give a default definition of a macro or a default value of an option. The definition will then be placed on the same line flush right. The star prevents the insertion of `\newline` after it. The optional bang adds the information that an option is mandatory, *i. e.*, it has to be set.

`\Module*!{`⟨*name*⟩`}`
Introduced in version 0.3
This command can be placed after `\option` but before `\Default` in order to determine the module the option belongs to. It will be written in the left margin next to the option name. The star prevents the insertion of `\newline` after it. The optional bang *adds* an index entry for the module. This is somehow inconsistent with many of the other commands where an optional star *prevents* an index entry but it fits to the functionality of `\Default` which is why this syntax was chosen.

The following demonstrates how the commands would be used to create option descriptions:

```
\begin{options}
  \opt{foo}
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \opt*{foo}\Default{bar}
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \opt{foo}\Module{bar}
    This option belongs to \module*{bar}.  Let's add a few more words so
    that the line gets filled and we can see how the output actually
    looks.
```

```
  \opt{foo}\Module{bar}\Default{baz}
    This option belongs to \module*{bar}.  Let's add a few more words so
    that the line gets filled and we can see how the output actually
    looks.
  \keyval{foo}{bar}\Default
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \keyval{foo}{bar}\Default!
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \keyval*{foo}{bar}
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \keyval-{foo}{bar}
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \keychoice{foo}{one,two,three}
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
  \keybool{foo}
    This makes stuff.  Let's add a few more words so that the line gets
    filled and we can see how the output actually looks.
\end{options}
```

The code above gives the following output:

`foo`
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` Default: `bar`
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`bar` » `foo`
This option belongs to the module `bar`. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`bar` » `foo` Default: `baz`
This option belongs to the module `bar`. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

`foo` = {⟨*bar*⟩} (initially empty)
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

foo = {⟨*bar*⟩} (required)
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

foo = {⟨*bar*⟩}
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

foo = ⟨*bar*⟩
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

foo = one|two|three
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

foo = true|false
This makes stuff. Let's add a few more words so that the line gets filled and we can see how the output actually looks.

8.3. Environment Descriptions

Environment descriptions are made – unsurprisingly – with the `environments` environment. It knows the command `\environment`:

`\environment*{`⟨*name*⟩`}[`⟨*stuff after*⟩`]`
This macro prints the environment name and puts a line break after it. The optional argument allows printing things directly after the environment name and can thus be used for adding arguments.

```
\begin{environments}
  \environment*{foobar}[\oarg{options}]
    This is environment \env*{foobar}.  The star prevents it from being
    added to the index.
\end{environments}
```

`\begin{foobar}[`⟨*options*⟩`]`
This is environment `foobar`. The star prevents it from being added to the index.

8.4. Code Examples

Code examples can be included through the `example` environment or the `sourcecode` environment. The `sourcecode` only shows the piece of LaTeXcode while the `example` environment also shows the output of the LaTeX code.

```
\begin{example}
  a \LaTeX\ code example
\end{example}
```

This example would give:

```
a \LaTeX\ code example
```

a LaTeX code example

Both environments can be influenced by options:

`code-only` = `true`|`false` Default: `false`

Only typeset the code as code but don't include it afterwards. The code box above is an example for the usage of this option. This option has no effect on the `sourcecode` environment: is is already set for this environment.

`side-by-side` = `true`|`false` Default: `false`

Typeset source and output side by side. The code is input on the left and the output on the right. Side by side examples are typeset in `minipage` environments with all consequences that come with them (think of `\parindent`, page breaks ...). Since a `minipage` cannot be broken across pages the surrounding mdframed frame gets the option `nobreak` = `true`. This option has no effect on the `sourcecode` environment.

`code-left` = `true`|`false` Default: `true`

If `true` and the option `side-by-side` is chosen the source code is printed on the right side else on the left. This option has no effect on the `sourcecode` environment.

`code-sep` = {⟨*definition*⟩} Default: `\hrulefill`

Code that is inserted between a source code and the corresponding output when printed below each other. This option has no effect on the `sourcecode` environment.

`outside` = `true`|`false` Default: `false`

Introduced in version 0.10

If `true` the output of an example is put outside of the frame in the input stream. This can be useful if the example code contains a floating environment for example.

The same example again, this time using `side-by-side` (which is the same as using the `sidebyside` environment):

```
1 a \LaTeX\ code example
```

a LaTeX code example

`side-by-side` and `code-left` = `false`:

a LaTeX code example

```
1 a \LaTeX\ code example
```

The frame around the examples is done by the mdframed package [DS13]. It is of course possible to customize it:

`add-frame-options` = {⟨*mdframed options*⟩} (initially empty)
Add options to the predefined settings.

`frame-options` = {⟨*mdframed options*⟩}
Default: `backgroundcolor=cnltxbg,linecolor=cnltx,roundcorner=5pt`
Overwrite the settings with new ones.

`add-local-frame` = {⟨*mdframed options*⟩}
Introduced in version 0.10
Add mdframed options to the environment where the option is used only. This is basically `\begin{mdframed}[style=cnltx,`⟨*options*⟩`]`.

`local-frame` = {⟨*mdframed options*⟩}
Introduced in version 0.10
replace the default mdframed options to the environment where the option is used only. This is basically `\begin{mdframed}[`⟨*options*⟩`]`.

The source code is formatted using the great listings package [HM19] by Carsten Heinz, Brooks Moses, and Jobst Hoffmann. Similar options exist to adapt listings' options that are used for formatting the source code. The predefined style has many options that will not be mentioned here. If you're interested you can find them in `cnltx-example.sty` or in section 11.2.1 on page 47.

`gobble` = ⟨*integer*⟩ Default: 2
The number of initial characters that is gobbled from each line.

`add-cmds` = {⟨*list of csnames*⟩} (initially empty)
A list of control sequence names that should be recognized as a command sequence in the source code examples and should be formatted accordingly. The control sequence names in this list will also get an index entry when they're used in the source example. This is done internally via `\csidx`. The option should be used to add the new commands that are defined by the package for which you are writing the manual for.

`add-silent-cmds` = {⟨*list of csnames*⟩}
A list of control sequence names that should be recognized as a command sequence in the source code examples and should be formatted accordingly. The control sequence names in this list will *not* get an index entry when they're used in the source example. There already is quite a large but far from comprehensive list of silent commands but many are still missing. This option allows you to extend the list on a per document basis.

`add-listings-options` = {⟨*listings options*⟩} (initially empty)
Additional options for the listings [HM19] environments. *This redefines the `cnltx` listings style which will affect all sourcecode environments!*

`listings-options` = {⟨*listings options*⟩}
Overwrite existing options with new ones. This can be used to build an own style from scratch. *This redefines the `cnltx` listings style which will affect all sourcecode environments!*

`add-sourcecode-options` = {⟨*listings options*⟩}
Introduced in version 0.4
These options are added to the listings options of the source code environments without redefing the main style. Hence it can be used to locally add options to a source code environment. This is basically `\lstset{style=cnltx,`⟨*options*⟩`}`.

`sourcecode-options` = {⟨*listings options*⟩}
Introduced in version 0.10
These options are added to the listings options of the source code environments without redefing or using the main style. Hence it can be used to locally add options to a source code environment. This is basically `\lstset{`⟨*options*⟩`}`.

`add-envs` = {⟨*list of environment names*⟩} (initially empty)
Like `add-cmds` but for environment names.

`add-silent-envs` = {⟨*list of environment names*⟩}
Like `add-silent-cmds` but for environment names.

8.5. Compile Source Examples

8.5.1. The Compliation Process

When you input an example like

```
\begin{example}
  \documentclass{article}
  \begin{document}
    foo
  \end{document}
\end{example}
```

you'll get an error since the code is input as is and you'll end up with `\documentclass` after `\begin{document}`. There's a way out, though.

Introduced in version 0.9

CNLTX-EXAMPLE provides the possibility to compile the source code file externally and input the compiled PDF.

```
\begin{example}[compile]
  \documentclass{article}
  \begin{document}
    foo
  \end{document}
\end{example}
```

This needs `shell-escape` enabled. The default compilation program is `pdflatex` which will compile the file two times. The process can be customized with the following options:

`compile` = `true`|`false` Default: `false`

Compile the source code file. Although this option can be set globally it really shouldn't be! It's best to give this option explicitly to the source code environment whose body should be compiled. If enabled globally *all* examples would be compiled and most likely lead to various errors since most examples won't be complete LaTeX documents.

`program` = `pdflatex`|`lualatex`|`xelatex`|`arara` Default: `pdflatex`

The program to compile the source file.

`runs` = {⟨*number*⟩} Default: `2`

The number of compilations.

`exe-with` = {⟨*options*⟩} (initially empty)

Command line options that can be given to the compilation program chosen with `program`.

`file-ext` = {⟨*extension*⟩} Default: `pdf`

The file extension of the included file of a compiled example.

`add-frame` = `true`|`false` Default: `true`

Introduced in version 0.10

If `true` every output page will get a frame.

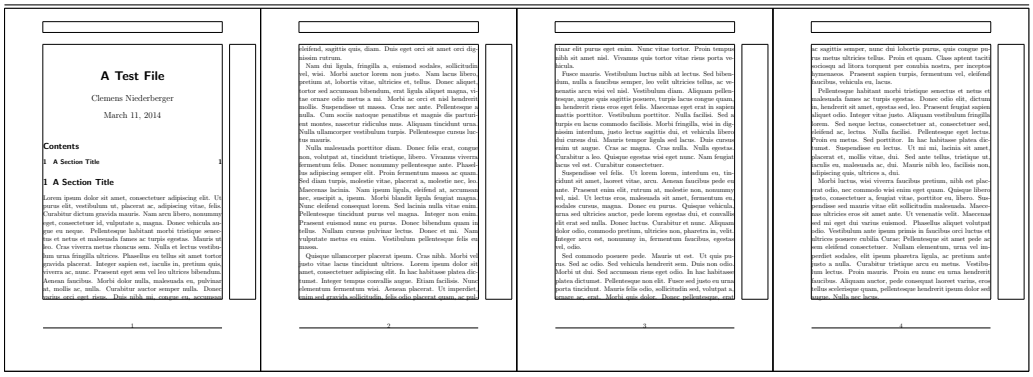
The compiled document will be input with `\includegraphics`, each page separately. Since the pages of the document are most likely as large as the ones from the main document itself they are scaled down. This is best demonstrated with an example. The following input

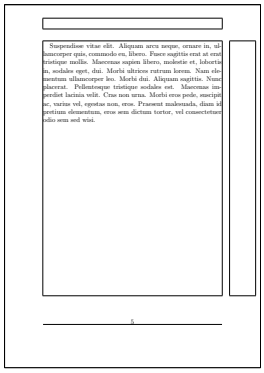
```
\begin{example}[compile]
  \documentclass[a5paper]{scrartcl}
  \usepackage{showframe,lipsum}
```

```
  \author{Clemens Niederberger}
  \title{A Test File}
  \begin{document}
    \maketitle
    \tableofcontents
    \section{A Section Title}
    \lipsum[1-10]
  \end{document}
\end{example}
```

will lead to this output:

```
\documentclass[a5paper]{scrartcl}
\usepackage{showframe,lipsum}
\author{Clemens Niederberger}
\title{A Test File}
\begin{document}
  \maketitle
  \tableofcontents
  \section{A Section Title}
  \lipsum[1-10]
\end{document}
```





The pages get scaled according to two parameters:

`max-pages` = {⟨*number*⟩} Default: 4
The maximum number of pages in a row. The width of the pages is scaled to `\linewidth`/n where n is either the number of pages p of the compiled document or ⟨*number*⟩ if $p >$ ⟨*number*⟩.

`max-height` = {⟨*dimension*⟩} Default: `.5\textheight`
The maximum height of a page.

There's another possibility to influence the appearance of the output:

`graphics` = {⟨*options*⟩} (initially empty)
⟨*options*⟩ are passed to `\includegraphics` for every page that is input.

8.5.2. Floating Output

Since the output can become a quite large figure it might be preferable to have it as a floating figure. This is also possible by using the option `float`.

`float` = `true`|`false`|⟨*float parameters*⟩ Default: `false`
Choose if the output should be placed in a `figure` of it's own. You can also use this option to specify the floating parameters for the float.

`float-pos` = {⟨*float parameters*⟩} Default: `tbp`
Set the standard floating parameters that are used if `float` = `true`. The default is actually the expansion of `\fps@figure` and not directly `tbp`.

`float-env` = {⟨*name*⟩} Default: `figure`
Introduced in version 0.10
The floating environment used when the option `float` is used.

`caption` = {⟨*text*⟩} (initially empty)
⟨*text*⟩ will be used as caption. If left blank no caption will be typeset. If you want to add a `\label` you can use it in this option. Implicitly sets `float` = `true`.

Please note that `float` only has an effect if `compile` = `true` has been set.

8.5.3. Selective Output

Sometimes it may be preferable not to include all pages of a compiled document but only specific pages. This is possible with the following option.

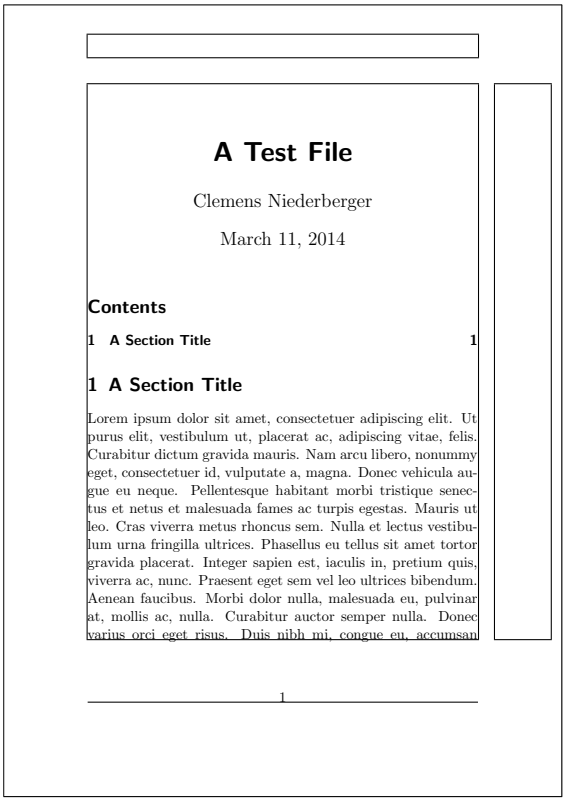
pages = {⟨*specifications*⟩}
Select the included pages. ⟨*specification*⟩ is a comma-separated list of page numbers and page ranges, *e. g.*, `1,3,4` or `1,3-5`. `1,3-5` is the same as `1,3,4,5`. If the list includes page numbers larger than the maximum number of pages the PDF has a warnung message will be issued and a replacement text will occur in the output where the page would have been.

The input

```
\begin{example}[compile,pages=1]
  \documentclass[a5paper]{scrartcl}
  \usepackage{showframe,lipsum}
  \author{Clemens Niederberger}
  \title{A Test File}
  \begin{document}
    \maketitle
    \tableofcontents
    \section{A Section Title}
    \lipsum[1-10]
  \end{document}
\end{example}
```

will lead to this output:

```
\documentclass[a5paper]{scrartcl}
\usepackage{showframe,lipsum}
\author{Clemens Niederberger}
\title{A Test File}
\begin{document}
  \maketitle
  \tableofcontents
  \section{A Section Title}
  \lipsum[1-10]
\end{document}
```



A Test File

Clemens Niederberger

March 11, 2014

Contents

1 A Section Title 1

1 A Section Title

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan

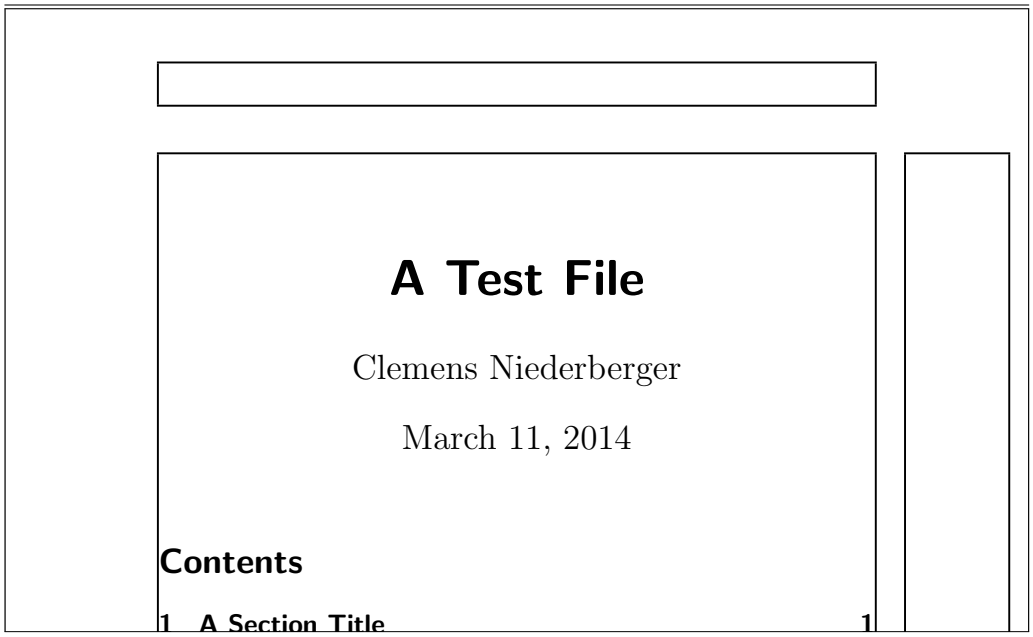
1

Together with the graphics option this can be used to output a part of a page. The following source

```
\begin{example}[compile,pages=1,graphics={trim={0pt 12cm 0pt 0pt},clip}]
  \documentclass[a5paper]{scrartcl}
  \usepackage{showframe,lipsum}
  \author{Clemens Niederberger}
  \title{A Test File}
  \begin{document}
    \maketitle
    \tableofcontents
    \section{A Section Title}
    \lipsum[1-10]
  \end{document}
\end{example}
```

will give this output:

```
\documentclass[a5paper]{scrartcl}
\usepackage{showframe,lipsum}
\author{Clemens Niederberger}
\title{A Test File}
\begin{document}
  \maketitle
  \tableofcontents
  \section{A Section Title}
  \lipsum[1-10]
\end{document}
```



8.6. Example File

Let's say you're documenting a package called `mypackage` that provides the command `\mycommand` and the environment `myenv`. The basic manual setup could then look something like this:

```
\documentclass[load-preamble]{cnltx-doc}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{mypackage}
\setcnltx{
  package  = mypackage ,
  authors  = John Doe ,
```

```
  email    = john@doe.com ,
  add-cmds = {mycommand} ,
  add-envs = {myenv}
}
\begin{document}
  ...
\end{document}
```

8.7. Additional Functionality Provided by CNLTX-BASE

The CNLTX-BASE package's main purpose is to provide programming facilities. Most of its macros are listed in section A.1. However, I like to explain some of its features in a bit more detail.

8.7.1. Looking for Trailing Punctuation

The command `\cnltx@ifpunctuation` is is a conditional that detects if a punctuaction mark follows and acts depending on it. What counts as a punctuation mark can be set by the user.

`\cnltx@ifpunctuation*[`⟨*punctuation marks*⟩`]{`⟨*true*⟩`}{`⟨*false*⟩`}`⟨*trailing punctuation*⟩

The starred version does not gobble the trailing punctuation while the unstarred does. That's why in the unstarred version you can also use `\cnltx@trailpunct` to access the gobbled punctuation mark. The optional argument sets the punctuation marks that should be considered for this use only.

`set-trail-punct = {`⟨*punctuation marks*⟩`}` Default: `,.!?;:`

Sets the default list of punctuation marks that should be checked if the optional argument of `\cnltx@ifpunctuation` is not used.

The usage is probably self-explaining:

```
\makeatletter
\cnltx@ifpunctuation{(test\cnltx@trailpunct)}{(test)}!\par
\cnltx@ifpunctuation[.]{(test\cnltx@trailpunct)}{(test)}!\par
a punctuation mark \cnltx@ifpunctuation*{follows}{doesn't follow}!\par
a full stop \cnltx@ifpunctuation*[.]{follows}{doesn't follow}!
```

(test!)
(test)!
a punctuation mark follows!
a full stop doesn't follow!

If the non-starred variant has gobbled a \par the \par is placed back:

```
\makeatletter
\def\test{\cnltx@ifpunctuation{(test\cnltx@trailpunct)}{(test)}}%
\makeatother
\test

\test.

\test{} .
```

(test)
(test.)
(test) .

8.7.2. Counter Representation Commands

Background
A counter representation command like \arabic{section} always is a command that calls an associated internal command (\@arabic in the case of our example) that acts on the count associated with the counter:

```
\def\arabic#1{\expandafter\@arabic\csname c@#1\endcsname}
\def\@arabic#1{\number #1}
```

The command \arabic{⟨*counter*⟩} builds a command sequence \c@⟨*counter*⟩ from its argument ⟨*counter*⟩. It then calls the internal command \@arabic that takes this command sequence as an argument. The command sequence \c@⟨*counter*⟩ is the count (in the TEX sense) that is associated with the counter ⟨*counter*⟩, *i. e.*, it holds the actual number. The command \@arabic now simply typesets the integer value of the count.

The same holds for every counter representation command. The principle always is as follows:

```
\def\foo#1{\expandafter\@foo\csname c@#1\endcsname}
\def\@foo#1{do something with #1 (where #1 is a count)}
```

This means in order to get a new counter representation command you actually need to define

two macros.

CNLTX-BASE defines an interface that allows to define both commands at once without having to think about \expandafter, associated counts, internal command names and so on. The only thing left to do is choosing a name for the counter representation and providing a valid definition of what should happen with the (integer) value of the counter.

New Commands

\DeclareCounterRepresentation{⟨*command*⟩}{⟨*definition*⟩}

Declares a new counter representation command and its internal equivalent. In the ⟨*definition*⟩ #1 is used to refer to the counter *number*, that is, the value of \c@⟨*counter*⟩. This command will silently overwrite any existing definition.

\newcounterrepresentation{⟨*command*⟩}{⟨*definition*⟩}

Defines a new counter representation command and its internal equivalent. In the ⟨*definition*⟩ #1 is used to refer to the counter *number*, that is, the value of \c@⟨*counter*⟩. This command will issue an error if either the user command or the internal command (*cf.* \arabic and \@arabic) already exist.

\providecounterrepresentation{⟨*command*⟩}{⟨*definition*⟩}

Provides a new counter representation command and its internal equivalent. In the ⟨*definition*⟩ #1 is used to refer to the counter *number*, that is, the value of \c@⟨*counter*⟩. This command will define the commands only if neither the user command nor the internal command (*cf.* \arabic and \@arabic) already exist and will do nothing if either of them exist.

\renewcounterrepresentation{⟨*command*⟩}{⟨*definition*⟩}

Redefines an existing counter representation command and its internal equivalent. In the ⟨*definition*⟩ #1 is used to refer to the counter *number*, that is, the value of \c@⟨*counter*⟩. This command will issue an error if neither the user command nor the internal command (*cf.* \arabic and \@arabic) already exist.

Let's take a look at what is actually defined by these commands:

```
\makeatletter\ttfamily
before:\par
\meaning\arabic\par
\meaning\@arabic

\renewcounterrepresentation\arabic{\the\numexpr#1\relax}%
after:\par
\meaning\arabic\par
\meaning\@arabic
```

```
before:
macro:#1->\expandafter \@arabic \csname c@#1\endcsname
```

```
macro:#1->\number #1
after:
macro:#1->\expandafter \@arabic \csname c@#1\endcsname
macro:#1->\the \numexpr #1\relax
```

As you can see nothing bad happens. The commands are only a convenient interface. Let's take a look at some more realistic examples. The above redefinition was only a demonstration. For example you may want to have a representation which calculates the displayed value from the counter value?

```
\newcounterrepresentation\minusone{\the\numexpr#1-1\relax}%
\newcounterrepresentation\multoffourrm{\romannumeral\numexpr(4*#1)-4\relax}%
% \newrobustcmd is provided by the `etoolbox' package
\newrobustcmd*\circlenumber[1]{%
  \tikz[baseline]\node[anchor=base,draw,shape=circle]{\number#1};}%
\newcounterrepresentation\circled{\circlenumber{#1}}%
\makeatletter
\newcounterrepresentation\twodigits{\two@digits{#1}}%
\makeatother
\newcounter{test}%
\setcounter{test}{9}

\minusone{test}\par
\multoffourrm{test}\par
\circled{test}\par
\twodigits{test}
```

8
xxxii
9
09

8.7.3. Expandable Document Commands

The commands presented in this section are highly experimental. *Use them* only *if you really have to!*

`\newexpandablecmd*{`⟨*cs*⟩`}[`⟨*num args*⟩`][`⟨*default opt*⟩`]{`⟨*definition*⟩`}`

Introduced in version 0.7

This command has the same syntax as `\newcommand`. The difference is that if ⟨*cs*⟩ is defined with an optional argument it is still fully expandable. This comes with a cost: in order to still being able to check for the optional argument it needs to see a following token as argument. If it is used without optional argument and has no mandatory arguments it may be necessary

to add a trailing \empty or something. There's another drawback: a command \test thus defined cannot distinguish between \test[] and \test{[]} and will misinterpret the second as a present optional argument.

My recommendation is to never use this for defining a user command.[20] Use it in code you can control and only if you have to.

If you define a command *without* optional argument this command falls back to \newcommand.

\renewexpandablecmd*{⟨*cs*⟩}[⟨*num args*⟩][⟨*default opt*⟩]{⟨*definition*⟩}

Introduced in version 0.7

The equivalent of \renewcommand. See description of \newexpandablecmd for further details.

\provideexpandablecmd*{⟨*cs*⟩}[⟨*num args*⟩][⟨*default opt*⟩]{⟨*definition*⟩}

Introduced in version 0.7

The equivalent of \providecommand. See description of \newexpandablecmd for further details.

8.8. Additional Functionality Provided by CNLTX-TOOLS

8.8.1. Commands for Defining Different Document Macros

The CNLTX-TOOLS package defines some additional macros which provide useful functionality also in contexts *not* documenting a LaTeX package.

\newname{⟨*cs*⟩}{⟨*first name*⟩ ⟨*last name*⟩}

Changed in version 0.12

Defines ⟨*cs*⟩ to write out the full name and add an index entry sorted by the last name. Also defines a starred variant of ⟨*cs*⟩ that only writes the last name but still adds the full index entry.

\name*{⟨*first name*⟩ ⟨*last name*⟩}

Changed in version 0.12

Typesets a name according to the same specs as the names defined with \newname. Also adds the name to the index. The starred version only writes the name but doesn't add the name to the index. Index entries either have the form ⟨*last name*⟩ or ⟨*last name*⟩, ⟨*first name*⟩ depending on the usage of the optional argument. It's safer to define a dedicated macro with \newname to get consistent index entries.

\cnltxacronym{⟨*pdf and sort string*⟩}{⟨*acronym*⟩}

Typesets ⟨*acronym*⟩ with small caps and uses ⟨*pdf and sort string*⟩ as PDF string and for sorting the index entry that is added. This command was used to define \lppl and \ctan. *This is not intended as a replacement for packages like acro [Nie19] or glossaries [Tal19]!* In fact it is a "poor man's" solution that allows me not to require one of those packages.

\newabbr*{⟨*control sequence*⟩}{⟨*definition*⟩}

Defines the abbreviation ⟨*control sequence*⟩ with the definition ⟨*definition*⟩. The star argument prevents that a dot is added at the end of the definition. An error is raised if ⟨*control sequence*⟩ already exists.

\renewabbr*{⟨*control sequence*⟩}{⟨*definition*⟩}

Redefines the abbreviation ⟨*control sequence*⟩ with the definition ⟨*definition*⟩. The star argument prevents that a dot is added at the end of the definition. An error is raised if ⟨*control sequence*⟩ does not exist already.

20. I can see the contradiction here: if a command is no user command there is no need for an optional argument.

`\defabbr*{`⟨*control sequence*⟩`}{`⟨*definition*⟩`}`
Defines or overwrites the abbreviation ⟨*control sequence*⟩ with the definition ⟨*definition*⟩. The star argument prevents that a dot is added at the end of the definition.

`\cnltxtimeformat{`⟨*abbreviation*⟩`}` Default: `\textsc{\,#1}`
Used in some predefined abbreviations.

`\cnltxlatin{`⟨*abbreviation*⟩`}` Default: `\textit{#1}`
Used in some localization strings.

`acronym-format = {`⟨*definition*⟩`}` Default: `\scshape`
Formatting of the acronyms as typeset with `\cnltxacronym`.

`name-format = {`⟨*formatting commands*⟩`}` Default: `#1`
The formatting of names created with `\newname` or typeset with `\name`. Names typeset through the bibliography style `cnltx` are also formatted according to this option. ⟨*formatting commands*⟩ should contain `#1` for the actual name.

`last-name-format = {`⟨*formatting commands*⟩`}` Default: `\textsc{#1}`
The formatting of the last names created with `\newname` or typeset with `\name`. Names typeset through the bibliography style `cnltx` are also formatted according to this option. ⟨*formatting commands*⟩ should contain `#1` for the actual name.

`first-name-format = {`⟨*formatting commands*⟩`}` Default: `#1`
The formatting of first names created with `\newname` or typeset with `\name`. Names typeset through the bibliography style `cnltx` are also formatted according to this option. ⟨*formatting commands*⟩ should contain `#1` for the actual name.

A short example of the usage of `\newname` and `\cnltxacronym`:

```
\newname\carlisle{David Carlisle}%
\carlisle\ is a well-known member of the \LaTeX\ community.  \carlisle* is
the author of many packages such as \pkg*{longtable}.  Take a look in the
index where you'll find \carlisle* mentioned.

\lppl\ is defined as \cnltxacronym{LPPL}{lppl}.
```

David Carlisle is a well-known member of the LaTeX community. David Carlisle is the author of many packages such as longtable. Take a look in the index where you'll find David Carlisle mentioned.
lppl is defined as lppl.

8.8.2. Defining Abbreviations

In section 8.8.1 when describing \newabbr and similar commands I said "The star argument prevents that a dot is added at the end of the definition". We should clarify what that means. Many abbreviations end with a dot. Some don't which explains the starred form of the commands. But why add a dot automatically in the first place? The reasoning is two-fold:

- Suppose you add the dot explicitly in the definition but forget one or two times that you did – you'll end up with abbreviations followed by *two* dots! Macros defined with CNLTX-TOOLS recognize a following dot and will not print a second one in those cases.
- In a document where \nonfrenchspacing is active the space after a dot in the middle of a sentence should be shorter than the one after the full stop ending a sentence. TEX automatically interprets a dot following a small letter as the end of a sentence and a dot after a capital letter as a dot after an abbreviation inside of a sentence. Usually you solve this by adding \@ in the appropriate places: `e.\,g.\@` for a intra-sentence space and `NSA \@.` for a inter-sentence space. The dot added by CNLTX-TOOLS always will be followed by an intra-sentence space. If you add a dot explicitly it will be your responsibility. Per default it will then act like a dot after a small letter.

Let's see some example:

```
\ttfamily% <= this will amplify the
   visual effect of \nonfrenchspacing
\newabbr\ab{a.b}%
\newabbr\AB{A.B}%
\newabbr*\cd{cd}%
\ab\ and some words\par
\ab. and some words\par
\AB\ and some words\par
\AB. and some words\par
\cd\ and some words
```

```
a.b. and some words
a.b.  and some words
A.B. and some words
A.B.  and some words
cd and some words
```

Beware: CNLTX-TOOLS will only leave the dot out if one follows directly in the input! That means that spaces are not ignored. However, of course TEX ignores spaces after macro names so usually this won't be an issue. If you define an abbreviation with a macro name consisting of one non-letter where spaces are not ignored you have to keep this fact in mind, though.

8.8.3. Predefined Abbreviations

CNLTX-TOOLS already provides a bunch of abbreviations defined with its \newabbr command.

Abbreviations that allow Localization

CNLTX-TOOLS defines a few abbreviations that are sensitive to babel settings. Currently only translations for English and German are provided and the definition falls back to the English

version if you're using a language other than those. It is possible to add further localization strings quite easily, see section 14.

`\ie`
Prints "*i. e.*" or "d. h."

`\eg`
Prints "*e. g.*" or "z. B."

`\etc`
Prints "*etc.*" or "etc."

`\cf`
Prints "*cf.*" or "vgl."

All of these macros add a final dot followed by `\@` *except* if a dot *directly* follows the macro.

```
\eg\ and some following text\par
\eg, and some following text\par
\eg. and some following text\par
\selectlanguage{ngerman}
\eg\ and some following text\par
\eg, and some following text\par
\eg. and some following text
```

e. g. and some following text
e. g., and some following text
e. g. and some following text
z. B. and some following text
z. B., and some following text
z. B. and some following text

German Abbreviations
The following abbreviations are not sensitive to localization and are only of use in a German text. Although they're defined: *please* do not use abbreviations at the start of a sentence!

`\dsh`
Prints "d. h."

`\Dsh`
Prints "D. h."

`\usf`
Prints "usf."

`\usw`
Prints "usw."

`\uswusf`
Prints "usw. usf."

`\zB`
Prints "z. B."

`\ZB`
Prints “Z. B.”

`\vgl`
Prints “vgl.”

`\Vgl`
Prints “Vgl.”

These macros behave the same as the ones described in section 8.8.3 on page 33.

```
\dsh\ und weiterer Text\par
\dsh. und weiterer Text\par
\usw\ und weiterer Text\par
\usw. und weiterer Text\par
\usf\ und weiterer Text\par
\usf. und weiterer Text\par
\zB\ und weiterer Text\par
\zB. und weiterer Text
```

d. h. und weiterer Text
d. h. und weiterer Text
usw. und weiterer Text
usw. und weiterer Text
usf. und weiterer Text
usf. und weiterer Text
z. B. und weiterer Text
z. B. und weiterer Text

Time related Abbreviations

The abbreviations presented in this section differ from the others in that they’re formatted by the command `\cnltxtimeformat{}`, see section 8.8.1 on page 31.

`\AM`
Prints “ A.M.”

`\PM`
Prints “ P.M.”

`\AD`
Prints “ A.D.”

`\BC`
Prints “ B.C.”

In their current definition these abbreviations are meant to be used *directly* after the time of day or the date, respectively.

```
She left for work before 6\AM, but
did not arrive until 12\PM. The
interval 5\BC--5\AD\ is one year
shorter than the interval
95\AD--105\AD.
```

She left for work before 6 A.M., but did not arrive until 12 P.M. The interval 5 B.C.–5 A.D. is one year shorter than the interval 95 A.D.–105 A.D.

9. Formatting Possibilities

One of the goals I wanted to achieve with this package is a consistent look and an easy interface for customization. No font choice and no color choice is fixed. In this section ways to change the formatting are shown.

The formatting of the different commands provided by CNLTX and various other properties can be changed in two ways: either by redefining the internal commands that are used for the formatting or by setting a corresponding option. Both variants are described in the next subsections.

How the colors should be changed is described in section 13 on page 51.

9.1. Formatting by Redefining Hooks

You can change the formatting by redefining the following commands. They're all defined by the CNLTX-EXAMPLE package except where indicated differently.

`\codefont` Default: `\ttfamily`

This command is used for all formatting of source code.

`\sourceformat` Default: `\codefont\small`

Formatting of the listings.

`\exampleformat` (initially empty)

Special formatting of the output of a listing.

`\versionnoteformat` Default: `\footnotesize\sffamily\RaggedRight`

provided by CNLTX-DOC

Formatting of the notes introduced in section 6.2 on page 9.

`\packageformat` Default: `\sffamily`

The formatting of package names.

`\classformat` Default: `\sffamily`

The formatting of class names.

`\argumentformat` Default: `\normalfont\itshape`

The formatting of `\meta{⟨meta⟩}`.

```
1 \renewcommand*\codefont{\sffamily\bfseries}
2 \code{foo} and \cs*{bar}, option \option{baz}
```

---

foo and **\bar**, option **baz**

9.2. Formatting by Setting Options

You can change the formatting of by setting the following options. They're all defined by the CNLTX-EXAMPLE package except where indicated differently.

`title-format` = {⟨*definition*⟩} Default: `\bfseries\scshape`
Introduced in version 0.2
Formatting of the document title.

`abstract-width` = {⟨*dimension*⟩} Default: `.75\linewidth`
Introduced in version 0.6
The width of the `\parbox` the abstract as set with the `abstract` option is placed in.

`abstract-format` = {⟨*definition*⟩} Default: `\setlength\parskip{.333\baselineskip}`
Introduced in version 0.6
Code that is placed in the parbox the abstract is placed in *before* the abstract text.

`caption-font` = {⟨*definition*⟩} Default: `\normalfont\small\sffamily`
This option only has any effect if you use the option `load-preamble`, see section 10.5 on page 41 for details on the option.

`caption-label-font` = {⟨*definition*⟩} Default: `\normalfont\small\sffamily\scshape`
This option only has any effect if you use the option `load-preamble`, see section 10.5 on page 41 for details on the option.

`code-font` = {⟨*definition*⟩} Default: `\ttfamily`
Used for all formatting of source code.

`source-format` = {⟨*definition*⟩} Default: `\codefont\small`
Formatting of the listings.

`expl-format` = {⟨*definition*⟩} (initially empty)
Special formatting of the output of a listing.

`module-sep` = {⟨*definition*⟩} Default: `\,>>\,`
provided by CNLTX-DOC
Change the separator between module name and corresponding option name.

`version-note-format` = {⟨*definition*⟩} Default: `\footnotesize\sffamily\RaggedRight`
provided by CNLTX-DOC
Formatting of the notes introduced in section 6.2 on page 9.

`pkg-format` = {⟨*definition*⟩} Default: `\sffamily`
The formatting of package names.

`cls-format` = {⟨*definition*⟩} Default: `\sffamily`
The formatting of class names.

`arg-format` = {⟨*definition*⟩} Default: `\normalfont\itshape`
The formatting of `\meta{`⟨*meta*⟩`}`.

`default-format` = {⟨*code*⟩} Default: `\uline`
Introduced in version 0.2
The formatting of `\default`'s argument. ⟨*code*⟩'s last macro should take one argument.

```
\setcnltx{code-font=\sffamily\itshape}
\code{foo} and \cs*{bar}, option \option{baz}
```

foo and *\bar*, option *baz*

10. Commands, Options and Further Settings Directly Related to the CNLTX-DOC Class

10.1. Using Class Options

The CNLTX-DOC class only knows a few options:

`load-preamble` = `true`|`false` Default: `false`

See section 10.5 on page 41 for details.

`load-preamble+` = `true`|`false` Default: `false`

See section 10.6 on page 44 for details.

`add-index` = `true`|`false` Default: `false`

See section 10.6 on page 44 for details.

`adapt-layout` = `true`|`false` Default: `true`

Introduced in version 0.13

If set to true CNLTX-DOC will make some changes to the section and part formats, it will redefine the footnote format, set the header and footer and adapt the caption format.

`babel-options` = {⟨*options*⟩} Default: `english`

Options given to the babel[21] package. This option only has an effect if `load-preamble` = `true`.

`scrartcl` = {⟨*options*⟩} (initially empty)

Changed in version 0.13

Options that are passed to the underlying class scrartcl. Can be used to pass options to scrartcl when `load-preamble` is used. Not really necessary since unknown options are passed to scrartcl, anyway.

10.2. Information on the Described Package or Class

A manual for a package or a class needs some information on the described package like the package name, the version number, the date and so on. This information is given with the following options. They are used to build the title page of the manual.

`package` = {⟨*package*⟩}

The name of the package that is described. Either this option or `class` or `name` should always be given. This command also defines a command sequence from the package name that formats the package name with color and small caps like CNLTX.

21. on CTAN as `babel`: `http://mirrors.ctan.org/macros/latex/required/babel/`

`class` = {⟨*class*⟩}
The name of the class that is described. Either this option or `package` or `name` should always be given. This command also defines a command sequence from the class name that formats the class name with color and small caps like CNLTX.

`name` = {⟨*name*⟩}
The name of the class/package that is described. Either this option or `package` or `class` should always be given. This command also defines a command sequence from the class name that formats the class name with color and small caps like CNLTX.

`authors` = {⟨*author list*⟩}
Changed in version 0.4
Comma separated list of package/class authors. After each author name you can add an email address by writing it in square brackets: `Some Name[some@name.com]`. Email addresses specified this way get written as a footnote. At least one author should always be given.

`version` = {⟨*version number*⟩}
Version number of the package/class. CNLTX tries to extract the information from the given `package` or `class`. This option can be used to set it explicitly.

`date` = {⟨*date*⟩}
Date of the package/class. CNLTX tries to extract the information from the given `package` or `class`. This option can be used to set it explicitly.

`info` = {⟨*package/class info*⟩}
Information about the package/class. CNLTX tries to extract the information from the given `package` or `class`. This option can be used to set it explicitly.

`subtitle` = {⟨*subtitle*⟩}
A subtitle, printed below the package/class name.

`url` = {⟨*url*⟩}
The homepage of the package.

`email` = {⟨*email*⟩}
A contact email address.

`abstract` = {⟨*abstract*⟩}
An abstract of the package/class/manual. This is text typeset in a box of `.75\linewidth`. Actually it does not have to be text but could be an image or whatever you like.

10.3. Building of the Manuals Title Page

If either the `package` or `class` has been given an automatic title page is built using the gathered information. Figure 1 on the next page roughly sketches which informations is used and how the different elements are arranged on the title page. The page style of the title page is `plain`. Additionally a table of contents is automatically built that is set in two columns. The automatic building of the title page can be prevented by explicitly setting the following option:

`build-title` = `true`|`false`
The default state depends on other options given like `package`. However, setting this option to `false` *after* any of the options described in section 10.2 on page 38 will prevent the building of a title page and allows you to design your own.

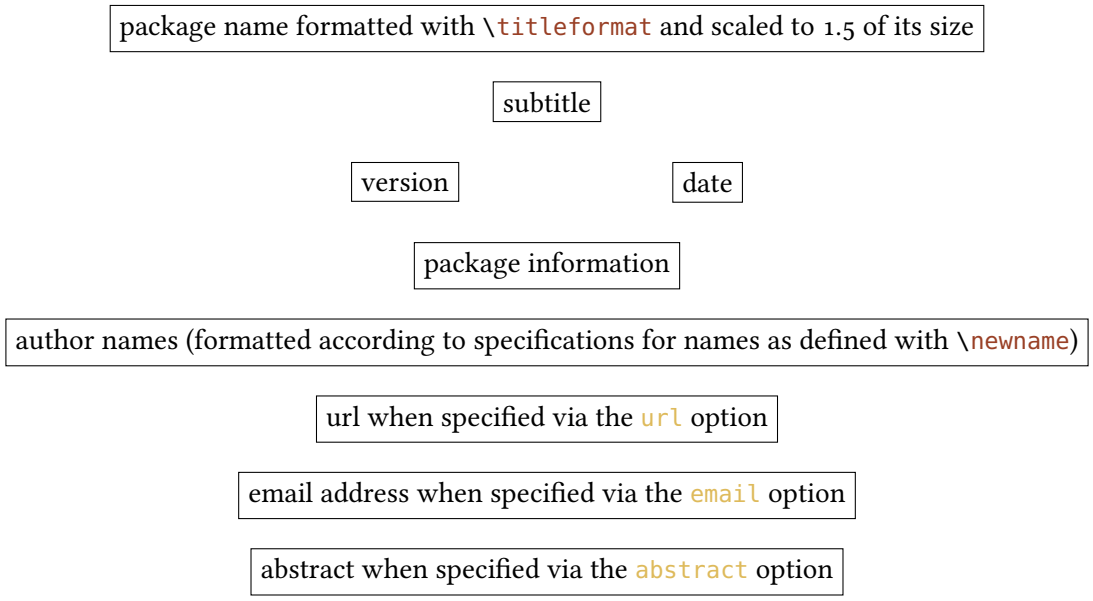


FIGURE 1: Schematic sketch of the title page.

10.4. A Quotation Environment

Introduced in version 0.5

CNLTX-DOC provides a quotation environment:

`\begin{cnltxquote}[`⟨*author/reference*⟩`]`
A quotation environment.

The environment sets the body indented on both sides as it simply uses a `quote` environment internally. The contents of the optional argument is set flush right after the environment's body. The formatting is controlled by two options:

`quote-format` = {⟨*definition*⟩} Default: `\small\sffamily`
The formatting of the environment's body.

`quote-author-format` = {⟨*definition*⟩} Default: `\itshape`

```
\begin{cnltxquote}[Douglas Adams, The Restaurant at the End of the Universe]
  ``The first ten million years were the worst,'' said Marvin, ``and the
  second ten million years, they were the worst too.  The third ten million
  years I didn't enjoy at all.  After that I went into a bit of a decline.''
\end{cnltxquote}
```

> "The first ten million years were the worst," said Marvin, "and the second ten million years, they were the worst too. The third ten million years I didn't enjoy at all. After that I went into a bit of a decline."
>
> *Douglas Adams, The Restaurant at the End of the Universe*

10.5. Predefined Preamble

It is possible to load a part of my standard preamble automatically by passing an option as class option.

`load-preamble`

Class option that preloads part of my custom preamble.

Changed in version 0.13

Using the option will include the following code:

```
\RequirePackage{ifxetex,ifluatex}
\ifbool{cnltx@load@fonts}
  {
    \ifboolexpr{not bool{xetex} and not bool{luatex}}
      {\RequirePackage[T1]{fontenc}}
      {\RequirePackage{fontspec}}
    \RequirePackage[oldstyle]{libertine}
    \RequirePackage{libertinehologopatch}
    \RequirePackage[supstfm=libertinesups]{superiors}
    % libertine does not have superior letters:
    \def\@makefnmark{%
      \hbox{%
        \cnltx@ifisnum{\@thefnmark}
          {\textsu{\hspace*{\superiors@spaced}\@thefnmark}}
          {\@textsuperscript{\normalfont\@thefnmark}}%
      }%
    }
  }
  {}
\ifbool{cnltx@microtype}
  {\RequirePackage{microtype}}
```

```
  {}
\ifboolexpr
  {
    bool {cnltx@microtype} and
    not test {\ifcsdef{MT@pr@set@@romansans}} and
    not test {\ifcsdef{MT@ex@set@@romansans}} and
    bool {cnltx@load@fonts}
  }
  {
    \DeclareMicrotypeSet{romansans}{
      encoding = {*},
      family   = {rm*,sf*}
    }
  }
  {}
\ifboolexpr
  {
    bool {cnltx@microtype} and
    not test {\ifcsdef{MT@tr@set@@scshape}} and
    bool {cnltx@load@fonts}
  }
  {
    \DeclareMicrotypeSet[tracking]{scshape}{
      encoding = {*} ,
      shape    = {sc,scit,si}
    }
  }
  {}
\ifboolexpr
  {
    not bool {xetex} and
    bool {cnltx@load@fonts} and
    bool {cnltx@microtype}
  }
  {
    \microtypesetup{
      tracking   = scshape ,
      protrusion = romansans ,
      expansion  = romansans
    }
    \DisableLigatures{ family = tt* }
  }
  {}
\ifbool{cnltx@load@fonts}
  {
    \ifboolexpr{not bool{xetex} and not bool{luatex}}
      {\RequirePackage[scaled=.81]{beramono}}
      {
```

```
      \setmonofont[
        Scale = MatchLowercase ,
        Ligatures = {NoCommon,NoRequired,NoContextual}
      ]{DejaVu Sans Mono}
    }
    \KOMAoptions{DIV=last}
    \recalctypearea
  }
  {}
\RequirePackage{fnpct}
\expandafter\RequirePackage\expandafter[\cnltx@babel@options]{babel}
```

The effect of this preamble is demonstrated by the document you're reading at this moment.

`load-preamble-` = `true`|`false` Default: `false`

This option has the same effect as adding using `load-preamble` but without making any font decisions.

Another option affects the layout of the document:

`adapt-layout` = `true`|`false` Default: `true`

Introduced in version 0.13

If set to true CNLTX-DOC will make some changes to the section and part formats, it will redefine the footnote format, set the header and footer and adapt the caption format.

Using the option will include the following code:

```
\renewcommand*\sectionformat{%
  \textcolor{cnltx}{\thesection\autodot}\enskip
}
\renewcommand*\subsectionformat{%
  \textcolor{cnltx}{\thesubsection\autodot}\enskip
}
\renewcommand*\subsubsectionformat{%
  \textcolor{cnltx}{\thesubsubsection\autodot}\enskip
}
\setkomafont{subsubsection}{\normalfont\normalsize\itshape}
\renewcommand*\partformat{%
  \textcolor{cnltx}{\partname~\thepart\autodot}%
}
\renewcommand*\partformat{%
  \textcolor{cnltx}{\partname~\thepart\autodot}}
\deffootnote{2em}{1em}{\llap{\thefootnotemark. }}%
\RequirePackage{scrlayer-scrpage}
\chead{\rightmark}
\KOMAoptions{automark}
\pagestyle{scrheadings}
```

```
\setcapindent{1.5em}
\setkomafont{caption}{\cnltx@caption@font}
\setkomafont{captionlabel}{\cnltx@captionlabel@font}
```

10.6. Predefined Indexing

CNLTX-DOC allows the automated creation of an index. This is done with the help of the imakeidx package by Enrico GREGORIO [Gre16]. To use this feature you have two class options. They cannot be set with `\setcnltx` but must be given as class options.

`add-index` = `true`|`false` Default: `false`
Enables the automatic creation of an index at the end of the document.

`load-preamble+` = `true`|`false` Default: `false`
This option has the same effect as adding the options `load-preamble`, `add-index` and `add-bib`.

Enabling the feature

- loads the imakeidx[22] package,
- uses a given style file for the index that can be specified with the `index-style` option,
- sets a certain setup for the index that can be specified with the `index-setup` option and
- adds an index at the end of the document.

The following options are available to customize the appearance of the index:

`index-prologue` = {⟨*text*⟩}
Adds ⟨*text*⟩ as index prologue between heading and the actual index.

`index-space` = {⟨*dimension*⟩} Default: `0pt`
The vertical space between index prologue and index.

`index-setup` = {⟨*options*⟩} Default: `othercode=\footnotesize,level=\addsec`
The options that are passed to imakeidx's `\indexsetup` command.

`makeindex-setup` = {⟨*options*⟩} Default: `columns=2,columnsep=1em`
The options that are passed to the `\makeindex` command.

`index-style` = {⟨*style file*⟩} Default: `cnltx.ist`
The style file that is used for formatting the index.

The index style file `cnltx.ist` contains the following lines:

22. on CTAN as `imakeidx`: `http://mirrors.ctan.org/macros/latex/contrib/imakeidx/`

```
heading_prefix "{\\bfseries "
heading_suffix "\\hfil}\\nopagebreak\n"
headings_flag  1
delim_0 "\\dotfill"
delim_1 "\\dotfill"
delim_2 "\\dotfill"
delim_r "\\nohyperpage{\\textendash}"
delim_t ""
suffix_2p "\\nohyperpage{\\,\\GetTranslation{cnltx-f.}\\@}"
suffix_3p "\\nohyperpage{\\,\\GetTranslation{cnltx-ff.}\\@}"
```

The feature is demonstrated by this document which does not contain a single control sequence containing the string `index`!

10.7. Bibliography with biblatex

10.7.1. Bibliography Entry Types `package`, `class` and `bundle` for biblatex

Introduced in version 0.4

CNLTX-DOC defines the bibliograpy entry types `package`, `class` and `bundle` when biblatex [Leh19a] is used. This allows specifying LaTeX packages in `bib` files:

```
@package{pkg:chngcntr,
  title      = {chngcntr} ,
  author     = {Peter Wilson} ,
  maintainer = {Will Robertson} ,
  date       = {2009-09-02} ,
  version    = {1.0a} ,
  url        = {http://mirror.ctan.org/macros/latex/contrib/chngcntr/}
}
@class{cls:exam,
  title      = {exam},
  author     = {Philip Hirschhorn},
  date       = {2015-05-07},
  version    = {2.5},
  url        = {http://mirror.ctan.org/macros/latex/contrib/exam/}
}
@bundle{bnd:koma-script,
  title          = {\KOMAScript} ,
  sorttitle      = {KOMA-Script} ,
  indextitle     = {\KOMAScript} ,
  indexsorttitle = {KOMA-Script} ,
  author         = {Markus Kohm},
  date           = {2015-07-02} ,
  version        = {3.18} ,
  url            = {http://mirror.ctan.org/macros/latex/contrib/koma-script/}
```

```
}
```

As you can see also an entry field `maintainer` is defined. For this to work you have to use the biblatex bibliography style `cnltx`. This style basically is a clone of the style `alphabetic` but defines the necessary additions for the `package`, `class` and `bundle` entry types and the `maintainer` entry field.

Along with the bibliography style a citation style `cnltx` is provided, again a clone of the `alphabetic` style. The only addition it makes is that indexing of maintainer names is enabled if biblatex's `indexing` option is used. The styles load CNLTX-EXAMPLE as it relies on definitions made by it.

This document uses the following call of biblatex:

```
\usepackage[
  backend=biber,
  style=cnltx,
  sortlocale=en_US,
  indexing=cite,
  useprefix]{biblatex}
\addbibresource{cnltx.bib}
```

Actually it let's CNLTX-DOC do it, see section 10.7.2 for details.

Just for the sake of the example I am going to cite the chngcntr package now [Wil18] so you can see both the bibliography entry and the indexed names of package, author and maintainer in the appendix.

10.7.2. Automatic Bibliography

CNLTX-DOC allows the automated creation of a bibliography.

`add-bib` = `true`|`false` Default: `false`

Enables the automatic creation of a bibliography at the end of the document.

`load-preamble+` = `true`|`false` Default: `false`

This option has the same effect as adding the options `load-preamble`, `add-index` and `add-bib`.

What this options does is including the following code:

```
\RequirePackage[
  backend=biber,
  style=cnltx,
  sortlocale=en_US,
```

```
  indexing=cite,
  useprefix]{biblatex}
\addbibresource{cnltx.bib}
\AtEndDocument{\printbibliography}
```

As you can see there's also a bibliography database file `cnltx.bib` that provides a yet small but growing number of package entries.

11. Predefined listings and mdframed Styles

11.1. mdframed

The source code environments (see section 8.4 on page 18) all get a frame with the help of the mdframed [DS13] package. For this a custom style is defined called `cnltx`. The options `frame-options` and `add-frame-options` mentioned in section 8.4 on page 18 manipulate this style. It is predefined with these values:

```
\def\cnltx@mdframed@options{
  backgroundcolor = cnltxbg ,
  linecolor       = cnltx ,
  roundcorner     = 5pt
}
```

11.2. listings

11.2.1. LaTeX Sourcecode

The code of the source code environments (see section 8.4 on page 18) is formatted with the help of the listings package [HM19]. A listings style is defined called `cnltx`. The options `add-cmds`, `add-silent-cmds`, `add-envs`, `add-silent-envs`, `listings-options` and `add-listings-options` manipulate this style. It is predefined by CNLTX-EXAMPLE as follows:

```
\def\cnltx@listings@style{
  language         = [AlLaTeX]TeX,
  alsolanguage     = [plain]TeX,
  basicstyle       = {\sourceformat},
  numbers          = left,
  numberstyle      = \tiny,
  xleftmargin      = 1em,
  numbersep        = .75em,
```

```
gobble           = \cnltx@gobble ,
columns          = fullflexible,
literate         =
  {ä}{{\"a}}1
  {ö}{{\"o}}1
  {ü}{{\"u}}1
  {Ä}{{\"A}}1
  {Ö}{{\"O}}1
  {Ü}{{\"U}}1
  {ß}{{\ss}}1 ,
breaklines       = true,
keepspaces       = true,
breakindent      = 1em,
commentstyle     = \color{comment},
keywordstyle     = \color{cs},
deletetexcs      =
  {
    a,o,u,A,O,U,
    begin,
    center,
    description,document,
    end,enumerate,
    figure,flushleft,flushright,
    itemize,list,
    otherlanguage,
    table,tabu,tabular
  },
deletekeywords   =
  {
    a,o,u,A,O,U,
    begin,
    center,
    description,document,
    end,enumerate,
    figure,flushleft,flushright,
    itemize,list,
    otherlanguage,
    table,tabu,tabular
  },
% \begin, \end:
texcsstyle       = [2]\color{beginend},
index            = [2][texcs2],
indexstyle       = [2]\@gobble,
moretexcs        = [2]{begin,end},
% added environments that'll be indexed:
texcsstyle       = [3]\color{env},
index            = [3][texcs3],
indexstyle       = [3]\envidx,
```

```
  % environments that won't be indexed:
  texcsstyle       = [4]\color{env},
  index            = [4][texcs4],
  indexstyle       = [4]\@gobble,
  % control sequences that'll be indexed:
  texcsstyle       = [5]\color{cs},
  index            = [5][texcs5],
  indexstyle       = [5]\indexcs,
  % control sequences that won't be indexed:
  texcsstyle       = [6]\color{cs},
  index            = [6][texcs6],
  indexstyle       = [6]\@gobble
}
```

11.2.2. BibTeX Entries

Introduced in version 0.4

The CNLTX-LISTINGS package defines a listings language `BibTeX` that contains a huge number of bibentry types and bibentry field types, have a look at section 10.7.1 on page 45. CNLTX-EXAMPLE defines a listings style for formatting them called `cnltx-bibtex`:

```
\def\cnltx@bibtex@listings@style{
  language         = BiBTeX,
  basicstyle       = {\sourceformat},
  numbers          = left,
  numberstyle      = \tiny,
  xleftmargin      = 1em,
  numbersep        = .5em,
  gobble           = \cnltx@gobble ,
  columns          = fullflexible,
  literate         =
    {ä}{{\"a}}1
    {ö}{{\"o}}1
    {ü}{{\"u}}1
    {Ä}{{\"A}}1
    {Ö}{{\"O}}1
    {Ü}{{\"U}}1
    {ß}{{\ss}}1 ,
  breaklines       = true,
  keepspaces       = true,
  breakindent      = 1em,
  commentstyle     = \color{comment},
  keywordstyle     = \color{bibentry} ,
  keywordstyle     = [2]\color{bibentryfield}\itshape ,
  showstringspaces = false ,
}
```

11.2.3. makeindex Style Files

Introduced in version 0.7

CNLTX-LISTINGS defines a listings language `makeindex` that contains the keywords used in `makeindex` style files. CNLTX-EXAMPLE defines a listings style for formatting them called `cnltx-makeindex`:

```
\def\cnltx@makeindex@listings@style{
  language         = makeindex,
  basicstyle       = {\sourceformat},
  numbers          = left,
  numberstyle      = \tiny,
  xleftmargin      = 1em,
  numbersep        = .75em,
  gobble           = \cnltx@gobble ,
  columns          = fullflexible,
  literate         =
    {ä}{{\"a}}1
    {ö}{{\"o}}1
    {ü}{{\"u}}1
    {Ä}{{\"A}}1
    {Ö}{{\"O}}1
    {Ü}{{\"U}}1
    {ß}{{\ss}}1 ,
  breaklines       = true,
  keepspaces       = true,
  breakindent      = 1em,
  commentstyle     = \color{comment},
  keywordstyle     = \color{makeidxkey}\bfseries ,
  stringstyle      = \color{makeidxstring} ,
  showstringspaces = false
}
```

12. PDF Strings and hyperref

Since the formatting and indexing commands `\cs`, `\env`, `\option`, `\pkg`, `\cls` and `\key` are robust they are ignored in PDF strings. For this reason you should *only use the starred variants* in places where PDF bookmarks are built from such as section titles when you use hyperref [OR19]. Since CNLTX-DOC loads hyperref this means you should do so, too, when you use CNLTX-DOC. This is important for two reasons:

1. Indexing in strings that get written to the table of contents does noch make much sense, anyway, so the starred versions should be used in section titles even if you don't use hyperref.

2. When hyperref is loaded the mentioned commands are disabled in PDF strings in a way that *expects* them to be followed by a star. This means leaving the star out will result in `doesn't match its definition` errors.

13. Predefined Colors and Color-Schemes

13.1. Explicitly Defined Colors

The CNLTX-BASE package defines a number of colors:

`cnltxbrown`
Per default used for the control sequences.

`cnltxblue`
Per default used for module names.

`cnltxred`
Per default used as base color in various places.

`cnltxgreen`
Unused per default.

`cnltxgray`
Per default used for formatting comments.

`cnltxyellow`
Per default used for option names.

`cnltxformalblue`
Unused per default.

`cnltxformalred`
Unused per default.

13.2. Actual Used Color Names and Color Schemes

The colors defined in section 13.1 are not directly used with those names. Instead colors are used whose names describe their function rather than the color. For this the color names are mapped to actual colors and saved as a coloring scheme. There are currently three predefined color schemes whose definitions are given below. Those definitions also show the actually used color names. They are defined via the following command:

`\definecolorscheme{`⟨*name*⟩`}{`⟨*color assignments*⟩`}`
Introduced in version 0.5
Defines the color scheme ⟨*name*⟩. When used all assignments will be actually carried out with xcolor's `\colorlet` command. How to input ⟨*color assignments*⟩ will be immediately clear from the examples below.

To activate a color scheme for a document it is simply selected through an option:

`color-scheme` = {⟨*color scheme name*⟩} Default: `default`

Activate a color scheme previously defined with `\definecolorscheme`.

The 'default' color scheme is defined as follows:

```
\definecolorscheme{default}{
  cs            => cnltxbrown , % command sequences
  option        => cnltxyellow ,% options
  module        => cnltxblue ,  % modules
  comment       => cnltxgray ,  % comments
  beginend      => red ,        % \begin and \end
  env           => black ,      % environment names
  argument      => black ,      % argument delimiters
  meta          => black!80 ,   % arguments of \meta
  cnltx         => cnltxred ,   % base color
  cnltxbg       => white ,      % source code box background
  link          => black!90 ,   % hyperlinks
  versionnote   => black!75     % versioning notes text
  bibentry      => cnltxgreen , % BibTeX entry types
  bibentryfield => black,       % BibTeX entry fields
  expandable    => red ,        % the color used in \expandable
  unexpandable  => black ,      % the color used in \unexpandable
  makeidxkey    => cnltxgreen , % used for keywords in the cnltx-makeindex
                                % style
  makeidxstring => black        % used for strings in the cnltx-makeindex
                                % style
}
```

The 'blue' color scheme is defined this way:

```
\definecolorscheme{blue}{
  cs            => cnltxbrown ,
  option        => cnltxgreen ,
  module        => cnltxred ,
  comment       => cnltxgray ,
  beginend      => red ,
  env           => black ,
  argument      => black ,
  meta          => black!80 ,
  cnltx         => cnltxblue ,
  cnltxbg       => yellow!10 ,
  link          => cnltx ,
  versionnote   => black!75
```

```
  bibentry      => cnltxyellow ,
  bibentryfield => black ,
  expandable    => red ,
  unexpandable  => black ,
  makeidxkey    => cnltxyellow ,
  makeidxstring => black
}
```

Finally the 'formal' color scheme is defined like this:

```
\definecolorscheme{formal}{
  cs            => black ,
  option        => cnltxformalblue ,
  module        => cnltxblue ,
  comment       => cnltxgray ,
  beginend      => red ,
  env           => black ,
  argument      => black ,
  meta          => black!80 ,
  cnltx         => cnltxformalblue ,
  cnltxbg       => white ,
  link          => black!90 ,
  versionnote   => black!75 ,
  bibentry      => black ,
  bibentryfield => black ,
  expandable    => red ,
  unexpandable  => black ,
  makeidxkey    => black ,
  makeidxstring => black
}
```

14. Language Support

Introduced in version 0.2

The CNLTX-DOC, the CNLTX-EXAMPLE and the CNLTX-TOOLS package as well as the `cnltx.ist` index style and the `cnltx` biblatex style all rely on the translations package [Nie17] for providing some document language dependent strings.[23] Currently only translations for English and German are provided. Others can be added and the existing ones changed with the following commands provided by the translations package:

`\DeclareTranslation{`⟨*language*⟩`}{`⟨*keyword*⟩`}{`⟨*translation*⟩`}`

Define or redefine translations for the string identified by the ID ⟨*keyword*⟩.

23. Actually they depend on CNLTX-TRANSLATIONS which in turn loads translations.

`\RenewTranslation{`⟨*language*⟩`}{`⟨*keyword*⟩`}{`⟨*translation*⟩`}`
Renew translations for the string identified by the ID ⟨*keyword*⟩.

The strings defined by CNLTX are listed in table 1 on the following page. They are used in indexing strings and in different parts of the document.

Part III.
Appendix

A. Internal Helper Commands

The commands in this section are only described for the sake of completeness. They are not meant to be used in a document. Some of them might be useful in LaTeX programming, though. Expandable commands are marked with *.

A.1. Defined by CNLTX-BASE

Especially CNLTX-BASE defines some useful helper macros that are also used by the other packages and classes.

A.1.1. Related to the Bundle

* `\cnltx@@date`
The creation date of the current version of the bundle.

* `\cnltx@@version`
The version number of the bundle.

* `\cnltx@@info`
The short description of the bundle.

`\cnltx@create@bundle@message*{`⟨*module*⟩`}{Error|Warning|WarningNoLine|Info}`
Introduced in version 0.7
Create suiting error and warning messaging commands for the module ⟨*module*⟩ of the CNLTX bundle. The starred version creates messages for a class the un-starred version messages for a package.

`\cnltx@base@error{`⟨*message*⟩`}`
Issue an error message using `\PackageError{cnltx-base}`.

`\cnltx@base@warning{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarning{cnltx-base}`.

`\cnltx@base@warningnoline{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarningNoLine{cnltx-base}`.

TABLE 1: Overview over available internationalization key words.

Package/Class	key word	English version	German version
CNLTX-EXAMPLE	cnltx-package	package	Paket
CNLTX-EXAMPLE	cnltx-class	class	Klasse
CNLTX-EXAMPLE	cnltx-bundle	bundle	Bundle
CNLTX-EXAMPLE	cnltx-environment	environment	Umgebung
CNLTX-DOC	cnltx-default	Default	Voreinstellung
CNLTX-DOC	cnltx-empty	initially empty	zunächst leer
CNLTX-DOC	cnltx-required	required	erforderlich
CNLTX-DOC	cnltx-toc	Table of Contents	Inhaltsverzeichnis
CNLTX-DOC	cnltx-license	Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (http://www.latex-project.org/lppl.txt). The software has the status	Es ist erlaubt, diese Software unter den Bedingungen der LaTeX Project Public License (LPPL), Version 1.3 oder später, zu kopieren und zu verteilen (http://www.latex-project.org/lppl.txt). Sie hat den Status
CNLTX-DOC	cnltx-introduced	Introduced in version	Eingeführt in Version
CNLTX-DOC	cnltx-changed	Changed in version	Geändert in Version
CNLTX-DOC	cnltx-f.	f.	f.
CNLTX-DOC	cnltx-ff.	ff.	ff.
CNLTX-DOC	cnltx-maintainer	current maintainer	aktueller Maintainer
CNLTX-DOC	cnltx-maintainers	current maintainers	aktuelle Maintainer
CNLTX-TOOLS	cnltx-i.e.	*i. e*	d. h
CNLTX-TOOLS	cnltx-e.g.	*e. g*	z. B
CNLTX-TOOLS	cnltx-cf.	*cf*	vgl
CNLTX-TOOLS	cnltx-etc.	*etc*	etc

`\cnltx@base@info{`⟨*message*⟩`}`
Issue a message using `\PackageInfo{cnltx-base}`.

`\cnltx@define@colorscheme{`⟨*name*⟩`}{`⟨*scheme definition*⟩`}`
Command that can be used to define a color scheme.

`\cnltx@load@module{`⟨*CNLTX module*⟩`}`
Introduced in version 0.11
Loads the package `cnltx-`⟨*CNLTX module*⟩`.sty`.

`\cnltx@load@modules{`⟨*CNLTX modules*⟩`}`
Introduced in version 0.11
Maps the comma separated list ⟨*CNLTX modules*⟩ to `\cnltx@load@module`, leading and trailing spaces are trimmed.

A.1.2. Programming Tools

Message Handling

`\cnltx@create@message*{`⟨*prefix*⟩`}{`⟨*package/class name*⟩`}{Error | Warning | WarningNoLine | Info} {`⟨*detailed error message*⟩`}`
Changed in version 0.7
Create error and warning messaging commands `\`⟨*prefix*⟩`@error | warning | warningnoline | info{`⟨*message*⟩`}`. The starred version creates messages for a class the un-starred version messages for a package. All commands have one argument which takes the message. ⟨*prefix*⟩ will be all lowercase in the generated command.

`\cnltx@create@generic@message*{`⟨*prefix*⟩`}{`⟨*package/class name*⟩`}{Error|Warning|WarningNoLine|Info}`
Introduced in version 0.7
Create error and warning messaging commands `\`⟨*prefix*⟩`@error | warning | warningnoline | info{`⟨*message*⟩`}`. The starred version creates messages for a class the un-starred version messages for a package. All commands have one argument which takes the message *except for the error command which gets two arguments*, the first for the short version and the second for the detailed message. ⟨*prefix*⟩ will be all lowercase in the generated command.

Conditionals

* `\iftest{`⟨*test directive*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Introduced in version 0.7
Checks if ⟨*test directive*⟩ is true and either places ⟨*true*⟩ or ⟨*false*⟩ in the input stream. ⟨*test directive*⟩ should be a TEX test like `\ifx`⟨*token1*⟩⟨*token2*⟩, *i. e.*, demand an `\else` and `\fi`. This is a command in the spirit of etoolbox's `\ifbool` that does the same for a boolean ⟨*bool*⟩ defined with `\newif\if`⟨*bool*⟩ or `\newbool{`⟨*bool*⟩`}`. It corresponds to etoolbox's `test` directive for its `\ifboolexpr`.

* `\nottest{`⟨*test directive*⟩`}{`⟨*not true*⟩`}{`⟨*not false*⟩`}`
Introduced in version 0.7
Checks if ⟨*test directive*⟩ is not true and either places ⟨*not true*⟩ or ⟨*not false*⟩ in the input stream. Test directive should be a TEX test like `\ifx`⟨*token1*⟩⟨*token2*⟩, *i. e.*, demand an `\else` and `\fi`. This is a command in the spirit of etoolbox's `\notbool` that does the same for a boolean ⟨*bool*⟩ defined with `\newif\if`⟨*bool*⟩ or `\newbool{`⟨*bool*⟩`}`.

* `\cnltx@ifcounter{`⟨*counter*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.11

Checks if ⟨*counter*⟩ is a counter, *i. e.*, if the control sequence names `\c@`⟨*counter*⟩, `\cl@`⟨*counter*⟩, `\p@`⟨*counter*⟩ *and* `\the`⟨*counter*⟩ exist and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

`\cnltx@ifnextchars{`⟨*list of tokens*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`⟨*trailing token*⟩

Introduced in version 0.8

Tests if ⟨*trailing token*⟩ is any of those in ⟨*list of tokens*⟩ and either places ⟨*true*⟩ or ⟨*false*⟩ in the input stream without removing ⟨*trailing token*⟩.

`\cnltx@ifsym{`⟨*token*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

A generic version of LaTeX's `\@ifstar` that checks if ⟨*token*⟩ follows in the input stream. If yes it is removed and ⟨*true*⟩ is placed in the input stream else ⟨*false*⟩.

`\cnltx@ifdash{`⟨*true*⟩`}{`⟨*false*⟩`}`

A wrapper for `\cnltx@ifsym{-}`.

`\cnltx@ifbang{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.3

A wrapper for `\cnltx@ifsym{!}`.

* `\cnltx@ifisnum{`⟨*token list*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.6

Checks if ⟨*token list*⟩ is an integer zero or greater and leaves ⟨*true*⟩ in the input stream if it is and ⟨*false*⟩ if it isn't. There is one hopefully extremely unlikely case where the test fails: when ⟨*token list*⟩ starts with "⟨*integer*⟩`%`" where `%` has a category code different than 9 (ignored) or 14 (comment).

* `\cnltx@ifshellescape{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.9

Checks if shellescape is enabled. It returns true if pdftexcmds' `\pdf@shellescape` has the value 1. This is a wrapper for `\iftest{\ifnum\pdf@shellescape=1 }`.

`\cnltx@if@in{`⟨*tokenlist*⟩`}{`⟨*search*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Places ⟨*true*⟩ in the input stream if ⟨*search*⟩ is found in ⟨*tokenlist*⟩ and ⟨*false*⟩ if it isn't.

* `\cnltx@ifstrequal{`⟨*string1*⟩`}{`⟨*string2*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.10

Tests if ⟨*string1*⟩ is equal to ⟨*string2*⟩ and either leaves ⟨*true*⟩ or ⟨*false*⟩ in the input stream. This test doesn't take category codes into account.

`\cnltx@ifinlist{`⟨*item*⟩`}{`⟨*listmacro*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.10

A conditional for etoolbox lists similar to `\ifinlist` where braces in items are allowed. This wraps around the proposal in etoolbox's documentation to redefine `\do` and loop through the list.

`\cnltx@ifinlistcs{`⟨*item*⟩`}{`⟨*listcsname*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`

Introduced in version 0.10

A conditional for etoolbox lists similar to `\ifinlistcs` where braces in items are allowed. This wraps around the proposal in etoolbox's documentation to redefine `\do` and loop through the list.

Expansion Tools

`\expandtwice{`⟨*code*⟩`}`

Introduced in version 0.10

Expands ⟨*code*⟩ twice in an `\edef`-like context. This is a wrapper for `\unexpanded\expandafter\expandafter\expandafter`.

`\cnltx@expandargs(`⟨*specs*⟩`)`⟨*control sequence*⟩

Introduced in version 0.7

This is a LaTeX 2_ε version of expl3's `\exp_args:N`⟨*specs*⟩. The command expands the arguments of ⟨*control sequence*⟩ according to ⟨*specs*⟩. In ⟨*specs*⟩

- `N` means unexpanded token,
- `n` means unexpanded braced group,
- `c` means braced group converted into a control sequence name,
- `o` means braced group expanded once,
- `f` means braced group expanded with `\romannumeral`, and
- `x` means braced group expanded with `\edef`.

Category Code Stuff

`\cnltx@save@catcode{`⟨*token*⟩`}`

Introduced in version 0.11

Saves the current category code of ⟨*token*⟩.

`\cnltx@restore@catcode{`⟨*token*⟩`}`

Introduced in version 0.11

Restores the category code of ⟨*token*⟩ as previously saved with `\cnltx@save@catcode`.

`\cnltx@set@catcode{`⟨*token*⟩`}{`⟨*catcode*⟩`}`

Introduced in version 0.11

Sets the category code of ⟨*token*⟩ to ⟨*catcode*⟩. This is a wrapper for `\catcode'`⟨*token*⟩`=`⟨*catcode*⟩`\relax`.

`\cnltx@save@catcodes{`⟨*tokenlist*⟩`}`

Introduced in version 0.11

Maps `\cnltx@save@catcode` to all tokens in ⟨*tokenlist*⟩.

`\cnltx@restore@catcodes{`⟨*tokenlist*⟩`}`

Introduced in version 0.11

Maps `\cnltx@restore@catcode` to all tokens in ⟨*tokenlist*⟩.

`\cnltx@set@catcodes{`⟨*tokenlist*⟩`}{`⟨*catcode*⟩`}`

Introduced in version 0.11

Maps `\cnltx@set@catcode` to all tokens in ⟨*tokenlist*⟩, *i. e.*, all tokens get category code ⟨*catcode*⟩.

`\cnltx@make@letter{`⟨*token*⟩`}`

Introduced in version 0.11

A wrapper for `\cnltx@set@catcode{`⟨*token*⟩`}{11}`.

`\cnltx@make@other{`⟨*token*⟩`}`

Introduced in version 0.11

A wrapper for `\cnltx@set@catcode{`⟨*token*⟩`}{12}`.

`\cnltx@make@active{`⟨*token*⟩`}`

Introduced in version 0.11

A wrapper for `\cnltx@set@catcode{`⟨*token*⟩`}{13}`.

Token List Manipulation

`\cnltx@replace@once{`⟨*cs*⟩`}{`⟨*search*⟩`}{`⟨*replace*⟩`}`
Replaces the first occurence of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩ with ⟨*replace*⟩.

`\cnltx@greplace@once{`⟨*cs*⟩`}{`⟨*search*⟩`}{`⟨*replace*⟩`}`
Introduced in version 0.9
The same as `\cnltx@replace@once` but acts globally.

`\cnltx@replace@all{`⟨*cs*⟩`}{`⟨*search*⟩`}{`⟨*replace*⟩`}`
Replaces all occurences of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩ with ⟨*replace*⟩.

`\cnltx@greplace@all{`⟨*cs*⟩`}{`⟨*search*⟩`}{`⟨*replace*⟩`}`
Introduced in version 0.9
The same as `\cnltx@replace@all` but acts globally.

`\cnltx@remove@once{`⟨*cs*⟩`}{`⟨*search*⟩`}`
Introduced in version 0.3
Removes the first occurence of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩.

`\cnltx@gremove@once{`⟨*cs*⟩`}{`⟨*search*⟩`}`
Introduced in version 0.9
The same as `\cnltx@remove@once` but acts globally.

`\cnltx@remove@all{`⟨*cs*⟩`}{`⟨*search*⟩`}`
Introduced in version 0.3
Removes all occurences of ⟨*search*⟩ in the first expansion of ⟨*cs*⟩.

`\cnltx@gremove@all{`⟨*cs*⟩`}{`⟨*search*⟩`}`
Introduced in version 0.9
The same as `\cnltx@remove@all` but acts globally.

Miscellaneous

* `\cnltx@par`
Expands to `\par`. Sometimes you need to smuggle a `\par` in a short macro …

* `\cnltx@stripbs`
A shortcut for `\expandafter\@gobble\string`.

`\cnltxat`
Robust command that typesets '@' with category code 11. An @ in command names confuses the indexing of the command names. Either one uses another symbol for makeindex's "actual" recognition and also tells idxcmds [Nie15] about it or one uses `\cnltxat` in `\cs` and friends. For the sake of convenience you can define a command like `\at` that expands to it.[24] In order not to overwrite any such existing macro it is not defined by CNLTX-EXAMPLE. This document for example defines `\def\at{\cnltxat}`.

`\cnltxletterat`
An alias of `\cnltxat`.

24. This is important. If you `\let` it to `\cnltxat` index entries may be sorted differently! Remember: `\cnltxat` is robust.

`\cnltxotherat`
The same as `\cnltxat` but with a '@' with category code 12.

`\cnltxbang`
The same as `\cnltxotherat` except that it contains a '!'.

`\cnltxequal`
The same as `\cnltxotherat` except that it contains a '='.

A.2. Defined by CNLTX-DOC

`\cnltx@doc@error{`⟨*message*⟩`}`
Issue an error message using `\ClassError{cnltx-doc}`.

`\cnltx@doc@warning{`⟨*message*⟩`}`
Issue a warning message using `\ClassWarning{cnltx-doc}`.

`\cnltx@doc@warningnoline{`⟨*message*⟩`}`
Issue a warning message using `\ClassWarningNoLine{cnltx-doc}`.

`\cnltx@doc@info{`⟨*message*⟩`}`
Issue a message using `\ClassInfo{cnltx-doc}`.

`\cnltx@getfileinfo{`⟨*file name*⟩`}{`⟨*file extension*⟩`}`
Extract the date, version and background information for a package or a class and defines `\cnltx@package@date`, `\cnltx@package@version` and `\cnltx@package@info` to contain the extracted data.

`\cnltx@version@note{`⟨*note*⟩`}`
Command that is used for the versioning notes interally. Sets `\reversemarginpar` and then writes the note ⟨*note*⟩ to the margin with corresponding formatting.

`\begin{cnltxlist}`
The list environment that is used by the environments `commands`, `options` and `environments`.

A.3. Defined by CNLTX-EXAMPLE

`\cnltx@example@error{`⟨*message*⟩`}`
Issue an error message using `\PackageError{cnltx-example}`.

`\cnltx@example@warning{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarning{cnltx-example}`.

`\cnltx@example@warningnoline{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarningNoLine{cnltx-example}`.

`\cnltx@example@info{`⟨*message*⟩`}`
Issue a message using `\PackageInfo{cnltx-example}`.

`\cnltx@isvalue`
Used in definitions of the key/value option typesetting commands. Inserts a = with some stretchable space around and a legal break-point after it.

`\indexcs`
Version of `\csidx` that takes care of a `\textcompwordmark` inserted by listings. Also replaces all occurences of @ with category code 11 or 12 with `\cnltxat`. Used to index commands in the `sourcecode` and `example` environments that have been added with `add-cmds`.

`\indexenv`
Introduced in version 0.7a
Version of `\envidx` that takes care of a `\textcompwordmark` inserted by listings. Also replaces all occurences of @ with category code 11 or 12 with `\cnltxat`. Used to index environments in the `sourcecode` and `example` environments that have been added with `add-envs`.

`\cnltx@treat@lst@index{`⟨*new index cs*⟩`}{`⟨*internal index cs*⟩`}`
Introduced in version 0.7a
This command was used to define `\indexcs` and `\indexenv`:
`\cnltx@treat@lst@index{\indexcs}{\csidx}`

`\MakePercentComment`
Sets the category code of % to 14.

`\cnltx@copyablespace`
Prints a space that is also copyable. Uses the accsupp package by Heiko Oberdiek [Obe18].

`\cnltx@mdframed@options`
Predefined option list for the mdframed [DS13] style `cnltx`.

`\cnltx@listings@style`
Predefined option list for the listings [HM19] style `cnltx`.

A.4. Defined by cnltx-listings

`\cnltx@listings@error{`⟨*message*⟩`}`
Issue an error message using `\PackageError{cnltx-listings}`.

`\cnltx@listings@warning{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarning{cnltx-listings}`.

`\cnltx@listings@warningnoline{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarningNoLine{cnltx-listings}`.

`\cnltx@listings@info{`⟨*message*⟩`}`
Issue a message using `\PackageInfo{cnltx-listings}`.

`\cnltx@predefined@control@sequences`
A comma-separated list of predefined ‘silent’ control sequence names.

`\cnltx@predefined@environments`
A comma-separated list of predefined ‘silent’ environment names.

`\listsilentcmds`
Prints all known control sequence names formatted and separated with the separator set with `list-sep`. Requires CNLTX-EXAMPLE.

`\listsilentenvs`
Prints all known environment names formatted and separated with the separator set with `list-sep`. Requires CNLTX-EXAMPLE.

`\listbibfilekeys{`⟨*file name*⟩`}`
Introduced in version 0.7
Prints all cite keys contained in the bibliography file ⟨*file name*⟩ formatted with `\code` and separated with the separator set with `list-sep`. Requires CNLTX-EXAMPLE.

`\listbibfiletypes{`⟨*file name*⟩`}`
Introduced in version 0.7
Prints all citation types contained in the bibliography file ⟨*file name*⟩ formatted with `\code` and separated with the separator set with `list-sep`. Requires CNLTX-EXAMPLE.

`\listbibfileentries{`⟨*file name*⟩`}`
Introduced in version 0.7
Prints all cite keys contained in the bibliography file ⟨*file name*⟩ formatted with `\code` and gives their respective entry types, separated with the separator set with `list-sep`. Requires CNLTX-EXAMPLE.

`list-sep` = `{`⟨*separator*⟩`}` Default: `,\space`
Sets the separator for CNLTX-LISTINGS' commands listing the different commands *etc.*

A.5. Defined by CNLTX-TOOLS

`\cnltx@tools@error{`⟨*message*⟩`}`
Issue an error message using `\PackageError{cnltx-tools}`.

`\cnltx@tools@warning{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarning{cnltx-tools}`.

`\cnltx@tools@warningnoline{`⟨*message*⟩`}`
Issue a warning message using `\PackageWarningNoLine{cnltx-tools}`.

`\cnltx@tools@info{`⟨*message*⟩`}`
Issue a message using `\PackageInfo{cnltx-tools}`.

`\cnltx@accsupp{`⟨*actual text*⟩`}{`⟨*additional options*⟩`}{`⟨*TEX text*⟩`}`
A wrapper for package accsupp's
`\BeginAccSupp{ActualText =` ⟨*actual text*⟩`}`⟨*TEX text*⟩`\EndAccSupp{}`.

B. List of Known LaTeX Control Sequences

Below all *predefined* control sequence names are listed that are treated as "silent" names by CNLTX, that is, those defined by CNLTX-LISTINGS.

\-, \@, \@alph, \@Alph, \@arabic, \@car, \@cdr, \@ctrerr, \@empty, \@firstofone, \@firstoftwo, \@gobble, \@ifclassloaded, \@ifnextchar, \@ifpackageloaded, \@ifstar, \@makefnmark, \@nil, \@roman, \@Roman, \@secondoftwo, \@slowromancap, \@textsuperscript, \@thefnmark, \a, \AA, \aa, \above, \abovedisplayshortskip, \abovedisplayskip, \abovewithdelims, \accent, \active, \acute, \addbibresource, \addcontentsline, \addpenalty, \addtocontents, \addtocounter, \addtolength, \addtokomafont, \addtoversion, \addvspace, \adjdemerits, \advance, \advancepageno, \ae, \AE, \afterassignment, \AfterEndPreamble, \AfterEndDocument, \AfterEndEnvironment, \aftergroup, \AfterPreamble, \aleph, \allocationnumber, \allowbreak, \alph, \Alph, \alpha, \amalg, \and, \angle, \approx, \appto, \apptocmd, \arabic, \arccos, \arcsin, \arctan, \arg, \arraycolsep, \arrayrulewidth, \arraystretch, \arrowvert, \Arrowvert, \ast, \asymp, \AtBeginDocument, \AtBeginDvi, \AtBeginEnvironment, \AtEndDocument, \AtEndEnvironment, \AtEndOfClass, \AtEndOfPackage, \AtEndPreamble, \atop, \atopwithdelims, \author, \author, \autodot, \b, \backslash, \badness, \bar, \baselineskip, \baselinestretch, \batchmode, \BeforeBeginEnvironment, \begingroup, \beginsection, \belowdisplayshortskip, \belowdisplayskip, \beta, \bezier, \bf, \bffam, \bfseries, \bgroup, \bibcite, \bibdata, \bibitem, \bibliography, \bibliographystyle, \bibstyle, \big, \Big, \bigbreak, \bigcap, \bigcirc, \bigcup, \bigg, \Bigg, \biggl, \Biggl, \biggm, \Biggm, \biggr, \Biggr, \bigl, \Bigl, \bigm, \Bigm, \bigodot, \bigoplus, \bigotimes, \bigr, \Bigr, \bigskip, \bigskipamount, \bigsqcup, \bigtriangledown, \bigtriangleup, \biguplus, \bigvee, \bigwedge, \binoppenalty, \bmod, \boldmath, \boolfalse, \booltrue, \bordermatrix, \bot, \botfigrule, \botmark, \bottomfraction, \bowtie, \Box, \box, \boxmaxdepth, \brace, \braceld, \bracelu, \bracerd, \braceru, \bracevert, \brack, \break, \breve, \brokenpenalty, \buildrel, \bullet, \bye, \c, \cal, \cap, \caption, \cases, \catcode, \cb, \cdot, \cdotp, \cdots, \centering, \centerline, \chapter, \char, \chardef, \check, \CheckCommand, \chi, \choose, \circ, \circle, \citation, \cite, \ClassError, \ClassInfo, \ClassWarning, \ClassWarningNoLine, \cleaders, \cleardoublepage, \clearpage, \cleartabs, \cline, \closein, \closeout, \clubpenalty, \clubsuit, \colon, \color, \columns, \columnsep, \columnseprule, \columnwidth, \cong, \contentsline, \coprod, \copy, \copyright, \cos, \cosh, \cot, \coth, \count, \countdef, \cr, \crcr, \cref, \csappto, \cseappto, \csgappto, \csxappto, \csc, \csdef, \csedef, \csgdef, \csxdef, \csdimdef, \csdimgdef, \csexpandonce, \csgluedef, \csgluegdef, \cslet, \csletcs, \csmudef, \csmugdef, \csname, \csnumdef, \csnumgdef, \cspreto, \csepreto, \csgpreto, \csxpreto, \csshow, \csundef, \csuse, \cup, \CurrentOption, \d, \dag, \dagger, \dashbox, \dashv, \date, \day, \dblfigrule, \dblfloatpagefraction, \dblfloatsep, \dbltextfloatsep, \dbltopfraction, \ddag, \ddagger, \ddot, \ddots, \deadcycles, \DeclareCharacterInheritance, \DeclareDictTranslation, \DeclareErrorFont, \DeclareFixedFont, \DeclareFontEncoding, \DeclareFontEncodingDefaults, \DeclareFontFamily, \DeclareFontShape, \DeclareFontSubstitution, \DeclareLanguage, \DeclareLanguageAlias, \DeclareLanguageDialect, \DeclareListParser, \DeclareMathAccent, \DeclareMathAlphabet, \DeclareMathAlphabet, \DeclareMathDelimiter, \DeclareMathRadical, \DeclareMathSizes, \DeclareMathSymbol, \DeclareMathVersion, \DeclareMicrotypeAlias, \DeclareMicrotypeBabelHook,

\DeclareMicrotypeSet, \DeclareMicrotypeSetDefault, \DeclareMicrotypeVariants, \DeclareOldFontCommand, \DeclareOption, \DeclarePreloadSizes, \DeclareRobustCommand, \DeclareSizeFunction, \DeclareSymbolFont, \DeclareSymbolFontAlphabet, \DeclareTextAccent, \DeclareTextAccentDefault, \DeclareTextCommand, \DeclareTextCommandDefault, \DeclareTextComposite, \DeclareTextCompositeCommand, \DeclareTextFontCommand, \DeclareTextSymbol, \DeclareTextSymbolDefault, \DeclareTranslation, \DeclareTranslationFallback, \def, \defaulthyphenchar, \defaultscriptratio, \defaultscriptscriptratio, \defaultskewchar, \defcounter, \deffootnote, \deffootnotemark, \definecolor, \deflength, \deg, \delcode, \delimiter, \delimiterfactor, \delimitershortfall, \delta, \Delta, \depth, \descriptionlabel, \det, \dh, \DH, \Diamond, \diamond, \diamondsuit, \dim, \dimdef, \dimgdef, \dimen, \dimendef, \dimexpr, \DisableLigatures, \discretionary, \displayindent, \displaylimits, \displaylines, \displaystyle, \displaywidowpenalty, \displaywidth, \div, \divide, \dj, \DJ, \do, \documentclass, \documentstyle, \dospecials, \dosupereject, \dot, \doteq, \dotfill, \dots, \doublehyphendemerits, \doublerulesep, \downarrow, \Downarrow, \downbracefill, \dp, \eappto, \edef, \egroup, \eject, \ell, \else, \em, \emergencystretch, \emph, \empty, \emptyset, \endcsname, \endgraf, \endgroup, \endinput, \endinsert, \enditemize, \endline, \endlinechar, \endlist, \endlrbox, \endmath, \endminipage, \endnote, \endpicture, \endsloppypar, \endtabbing, \endtabular, \endtrivlist, \endverbatim, \enlargethispage, \enskip, \enspace, \ensuremath, \epreto, \epsilon, \eqalign, \eqalignno, \eqno, \equiv, \errhelp, \errmessage, \errorcontextlines, \errorstopmode, \escapechar, \eta, \evensidemargin, \everycr, \everydisplay, \everyhbox, \everyjob, \everymath, \everypar, \everyvbox, \ExecuteOptions, \exhyphenpenalty, \exists, \exp, \expandafter, \expandonce, \extracolsep, \fam, \fbox, \fboxrule, \fboxsep, \fi, \filbreak, \filecontents, \fill, \finalhypendemerits, \firstmark, \fiverm, \fivebf, \fivei, \fivesy, \flat, \floatingpenalty, \floatpagefraction, \floatsep, \flushbottom, \fmtname, \fmtversion, \fnsymbol, \folio, \font, \fontdimen, \fontencoding, \fontfamily, \fontname, \fontseries, \fontshape, \fontsize, \fontspec, \fontsubfuzz, \footins, \footline, \footnote, \footnotemark, \footnoterule, \footnotesep, \footnotesize, \footnotetext, \footskip, \forall, \forlistloop, \foreignlanguage, \frac, \frame, \framebox, \frenchspacing, \frown, \fussy, \futurelet, \gamma, \Gamma, \gappto, \gcd, \ge, \GenericError, \GenericInfo, \GenericWarning, \geq, \gets, \GetTranslation, \GetTranslationFor, \gdef, \gg, \global, \globaldefs, \glossary, \gluedef, \gluegdef, \goodbreak, \gpreto, \grave, \H, \halign, \hang, \hangafter, \hangindent, \hat, \hbadness, \hbar, \hbox, \headheight, \headline, \headsep, \heartsuit, \height, \hfil, \hfill, \hfilneg, \hfuzz, \hglue, \hideskip, \hidewidth, \hline, \hoffset, \holdinginserts, \hom, \hookleftarrow, \hookrightarrow, \hphantom, \hrule, \hrulefill, \hsize, \hskip, \hskip, \hspace, \hss, \ht, \huge, \Huge, \hypersetup, \hyphenation, \hyphenchar, \hyphenpenalty, \i, \I, \ialign, \if, \ifblank, \ifbool, \ifboolexpe, \ifboolexpr, \ifcase, \ifcat, \ifcsdef, \ifcsname, \ifdim, \ifdef, \ifeof, \iff, \iffalse, \IfFileExists, \ifhbox, \ifhmode, \ifinlist, \ifinner, \ifmmode, \ifnum, \ifodd, \ifpatchable, \ifstr, \ifstrempty, \ifstrequal, \iftoggle, \iftrue, \ifvbox, \ifvmode, \ifvoid, \ifx, \ignorespaces, \ignorespacesafterend, \Im, \imath, \immediate, \in, \include, \includeonly, \indent, \inf, \infty, \indent, \index, \input, \InputIfFileExists,

\inputlineno, \insert, \insertpenalties, \int, \interdisplaylinepenalty, \interfootnotelinepenalty, \interlinepenalty, \intextsep, \intop, \iota, \it, \item, \itemindent, \itemitem, \itemize, \itemsep, \iterate, \itfam, \itshape, \j, \jmath, \jobname, \Join, \joinrel, \jot, \k, \kappa, \ker, \kern, \kill, \KOMAoption, \KOMAoptions, \l, \L, \label, \labelsep, \labelwidth, \labelenumi, \labelenumii, \labelenumiii, \labelenumiv, \labelitemi, \labelitemii, \labelitemiii, \labelitemiv, \lambda, \Lambda, \land, \langle, \language, \large, \Large, \LARGE, \lastbox, \lastkern, \lastpenalty, \lastskip, \LaTeX, \LaTeXe, \lbrace, \lbrack, \lccode, \lceil, \ldotp, \ldots, \le, \leaders, \leadsto, \leavevmode, \left, \leftarrow, \Leftarrow, \leftarrowfill, \lefteqn, \leftharpoondown, \leftharpoonup, \lefthyphenmin, \leftline, \leftmargin, \leftmargini, \leftmarginii, \leftmarginiii, \leftmarginiv, \leftmarginv, \leftmarginvi, \leftmark, \leftskip, \leftrightarrow, \Leftrightarrow, \leq, \leqalignno, \leqno, \let, \letcs, \lfloor, \limits, \linepenalty, \lineskip, \lineskiplimits, \lg, \lgroup, \lhd, \lhook, \lim, \liminf, \limsup, \line, \linebreak, \linespread, \linethickness, \linewidth, \list, \listadd, \listfiles, \listfiles, \listparindent, \ll, \llap, \lmoustache, \ln, \lnot, \LoadClassWithOptions, \LoadClass, \LoadDictionary, \LoadDictionaryFor, \LoadMicrotypeFile, \log, \long, \longleftarrow, \Longleftarrow, \longleftrightarrow, \Longleftrightarrow, \longmapsto, \longrightarrow, \loop, \looseness, \lor, \lower, \lowercase, \lq, \lslig, \lsstyle, \lstinline, \lstinputlisting, \lrbox, \ltx@ifnextchar, \LuaLaTeX, \LuaTeX, \mag, \magnification, \magstep, \magstephalf, \makeatletter, \makeatother, \makebox, \makefootline, \makeglossary, \makeheadline, \makeindex, \makelabel, \MakeLowercase, \maketitle, \MakeUppercase, \mapsto, \mapstochar, \marginpar, \marginparpush, \marginparsep, \marginparwidth, \mark, \markboth, \markright, \math, \mathaccent, \mathbf, \mathbin, \mathchar, \mathchardef, \mathchoice, \mathclose, \mathcode, \mathellipsis, \mathgroup, \mathhexbox, \mathinner, \mathit, \mathop, \mathopen, \mathord, \mathpalette, \mathparagraph, \mathpunct, \mathrel, \mathrm, \mathsection, \mathsf, \mathsterling, \mathstrut, \mathsurround, \mathtt, \mathunderscore, \mathversion, \matrix, \max, \maxdeadcycles, \maxdepth, \maxdimen, \mbox, \mdseries, \meaning, \medbreak, \medmuskip, \medskip, \medskipamount, \message, \MessageBreak, \mho, \microtypecontext, \microtypesetup, \mid, \midinsert, \min, \minipage, \mit, \mkern, \models, \month, \moveleft, \moveright, \mp, \mscount, \mskip, \mu, \mudef, \mugdef, \multicolumn, \multiply, \multiput, \multispan, \muskip, \muskipdef, \nabla, \narrower, \natural, \ne, \nearrow, \NeedsTeXFormat, \neg, \negthinspace, \neq, \newbox, \newbool, \newcommand, \newcount, \newcounter, \newdimen, \newenvironment, \newfam, \newfont, \newfontfamily, \newhelp, \newif, \newinsert, \newlabel, \newlanguage, \newlength, \newline, \newlinechar, \newmathalphabet, \newmuskip, \newpage, \newread, \newrobustcmd, \newsavebox, \newskip, \newtheorem, \newtoggle, \newtoks, \NewTranslation, \newwrite, \next, \ng, \NG, \ni, \noalign, \noboundary, \nobreak, \nobreakspace, \nocite, \nocorr, \nocorrlist, \node, \noexpand, \nofiles, \noindent, \nointerlineskip, \nolimits, \nolinebreak, \nonfrenchspacing, \nonscript, \nonstopmode, \nonumber, \nopagebreak, \nopagenumbers, \normalbaselines, \normalbaselineskip, \normalbottom, \normalcolor, \normalfont, \normalmarginpar, \normallineskip, \normallineskiplimit, \normalsize, \notblank, \notbool, \nottoggle, \nopagebreak, \not, \notin, \nu, \null,

\nulldelimiterspace, \nullfont, \number, \numberline, \numdef, \numgdef, \numexpr, \nwarrow, \o, \O, \oalign, \obeycr, \obeylines, \obeyspaces, \oddsidemargin, \odot, \oe, \OE, \of, \offinterlineskip, \oint, \ointop, \oldstyle, \oldstylenums, \omega, \Omega, \ominus, \omit, \onecolumn, \ooalign, \openin, \openout, \openup, \oplus, \OptionNotUsed, \or, \oslash, \otimes, \othersectionlevelsformat, \outer, \output, \outputpenalty, \oval, \over, \overbrace, \overfullrule, \overleftarrow, \overline, \overrightarrow, \overwithdelims, \owns, \P, \PackageError, \PackageInfo, \PackageWarning, \PackageWarningNoLine, \pagebody, \pagebreak, \pagecontents, \pagedepth, \pagefilllstretch, \pagefillstretch, \pagefilstretch, \pagegoal, \pageinsert, \pageno, \pagenumbering, \pageref, \pageshrink, \pagestretch, \pagestyle, \pagetotal, \paperheight, \paperwidth, \par, \paragraph, \paragraphmark, \parallel, \parbox, \parfillskip, \parindent, \parsep, \parshape, \parskip, \part, \partformat, \partial, \partname, \partopsep, \PassOptionsToClass, \PassOptionsToPackage, \patchcmd, \patterns, \pausing, \pdfLaTeX, \pdfstringdefDisableCommands, \pdfTeX, \penalty, \perp, \pgfkeys, \phantom, \phi, \Phi, \pi, \Pi, \picture, \plainoutput, \pm, \pmatrix, \pmod, \poptabs, \postdisplaypenalty, \pounds, \Pr, \prec, \preceq, \predisplaypenalty, \predisplaysize, \preloaded, \preto, \pretocmd, \pretolerance, \prevdepth, \prevgraf, \prime, \printacronyms, \printbibliography, \printendnotes, \printindex, \ProcessPgfOptions, \ProcessOptions, \proclaim, \prod, \propto, \protect, \protected, \protected@eappto, \protected@xappto, \protected@edef, \protected@xdef, \protected@epreto, \protected@xpreto, \protected@cseappto, \protected@csxappto, \protected@csedef, \protected@csxdef, \protected@csepreto, \protected@csxpreto, \protecting, \providebool, \providecommand, \providerobustcmd, \ProvidesClass, \ProvideDictionaryFor, \ProvidesFile, \ProvidesPackage, \ProvideTextCommand, \providetoggle, \psi, \Psi, \pushtabs, \put, \qbezier, \qbeziermax, \qquad, \quad, \r, \radical, \raggedbottom, \raggedleft, \RaggedLeft, \raggedright, \RaggedRight, \raise, \raisebox, \rangle, \rbrace, \rbrack, \rceil, \Re, \read, \recalctypearea, \ref, \refstepcounter, \relax, \relbar, \Relbar, \relpenalty, \removelastskip, \renewcommand, \renewenvironment, \renewrobustcmd, \RenewTranslation, \repeat, \RequirePackage, \RequirePackageWithOptions, \restorecr, \reversemarginpar, \rfloor, \rgroup, \rhd, \rho, \rhook, \right, \rightarrow, \Rightarrow, \rightarrowfill, \rightharpoondown, \rightharpoonup, \righthyphenmin, \rightleftharpoons, \rightline, \rightmargin, \rightmark, \rightskip, \rlap, \rm, \rmfamily, \rmoustache, \robustify, \roman, \Roman, \romannumeral, \root, \rootbox, \rq, \rule, \S, \samepage, \SaveTranslation, \SaveTranslationFor, \sb, \sbox, \scriptfont, \scriptscriptfont, \scriptscriptstyle, \scriptsize, \scriptspace, \scriptstyle, \scrollmode, \scshape, \searrow, \sec, \secdef, \section, \sectionmark, \selectfont, \selectlanguage, \setbool, \setbox, \setcapindent, \setcounter, \SetExpansion, \SetExtraKerning, \SetExtraSpacing, \setfnpct, \setkomafont, \setlanguage, \setlength, \setmainfont, \setmainlanguage, \SetMathAlphabet, \setminus, \setmonofont, \setotherlanguage, \setotherlanguages, \SetProtrusion, \setsansfont, \SetSymbolFont, \settabs, \settodepth, \settoggle, \settoheight, \settowidth, \SetTracking, \sevenbf, \seveni, \sevensy,

\sevenrm, \sfcode, \sffamily, \sharp, \shipout, \shorthandoff, \shorthandon, \shortstack, \show, \showbox, \showboxbreadth, \showboxdepth, \showhyphens, \showlists, \showoutput, \showoverfull, \showthe, \sidenote, \sigma, \Sigma, \sim, \simeq, \sin, \sinh, \skew, \skewchar, \skip, \skipdef, \sl, \slash, \slfam, \sloppy, \sloppypar, \slshape, \small, \smallbreak, \smallint, \smallskip, \smallskipamount, \smash, \smile, \sp, \space, \spacefactor, \spaceskip, \spadesuit, \span, \special, \splitbotmark, \splitfirstmark, \splitmaxdepth, \splittopskip, \sqcap, \sqcup, \sqrt, \sqsubset, \sqsubseteq, \sqsupset, \sqsupseteq, \ss, \SS, \stackrel, \star, \stepcounter, \stop, \stretch, \string, \strut, \strutbox, \subparagraph, \subparagraphmark, \subsection, \subsectionmark, \subset, \subseteq, \subsubsection, \subsubsectionmark, \succ, \succeq, \sum, \sup, \supereject, \suppressfloats, \supset, \supseteq, \surd, \swarrow, \symbol, \t, \tabalign, \tabbing, \tabbingsep, \tabcolsep, \tableofcontents, \tabskip, \tabular, \tabularnewline, \tabs, \tabsdone, \tabsyet, \tan, \tanh, \tau, \tenbf, \tenex, \teni, \tenit, \tenrm, \tensl, \tensy, \tentt, \TeX, \test, \textasciicircum, \textasciitilde, \textasteriskcentered, \textbackslash, \textbar, \textbf, \textbraceleft, \textbraceright, \textbullet, \textcircled, \textcompwordmark, \textcolor, \textdagger, \textdaggerdbl, \textdollar, \textellipsis, \textemdash, \textemdash, \textenglish, \textexclamdown, \textexclamdown, \textfloatsep, \textfont, \textfraction, \textgreater, \textindent, \textit, \textheight, \textless, \textls, \textmicrotypecontext, \textmd, \textnormal, \textogonekcentered, \textparagraph, \textperiodcentered, \textquestiondown, \textquotedbl, \textquotedblleft, \textquotedblright, \textquoteleft, \textquoteright, \textregistered, \textrm, \textsc, \textsection, \textsf, \textsl, \textsterling, \textstyle, \textsuperscript, \textsection, \textsubscript, \textsuperscript, \textsterling, \TextSymbolUnavailable, \texttt, \texttrademark, \textunderscore, \textup, \textvisiblespace, \textwidth, \th, \TH, \thanks, \the, \thechapter, \theendnotes, \theenumi, \theenumii, \theenumiii, \theenumiv, \thefootnote, \thefootnotemark, \thempfn, \thempfootnote, \thepage, \thepart, \thesection, \thesubsection, \thesubsubsection, \theta, \Theta, \thicklines, \thickmuskip, \thinlines, \thinmuskip, \thinspace, \thispagestyle, \tikz, \tilde, \time, \times, \tiny, \title, \to, \today togglefalse, \toggletrue, \toks, \toksdef, \tolerance, \top, \topfigrule, \topfraction, \topglue, \topins, \topinsert, \topmargin, \topmark, \topsep, \topskip, \totalheight, \tracingall, \tracingcommands, \tracingfonts, \tracinglostchars, \tracingmacros, \tracingonline, \tracingoutput, \tracingpages, \tracingparagraphs, \tracingpatches, \tracingrestores, \tracingstats, \triangle, \triangleleft, \triangleright, \trivlist, \tt, \ttfam, \ttfamily, \ttraggedright, \two@digits, \twocolumn, \typein, \typeout, \u, \uccode, \uchyph, \unboldmath, \undef, \underbar, \underbrace, \underline, \unhbox, \unhcopy, \unitlength, \unkern, \unlhd, \unpenalty, \unrhd, \unskip, \unvbox, \unvcopy, \uparrow, \Uparrow, \upbracefill, \updownarrow, \Updownarrow, \uplus, \uppercase, \upsilon, \Upsilon, \upshape, \usebox, \usecounter, \usefont, \UseMicrotypeSet, \usepackage, \UseTextAccent, \UseTextSymbol, \upshape, \v, \vadjust, \valign, \value, \varepsilon, \varphi, \varpi, \varrho, \varsigma, \vartheta, \vbadness, \vbox, \vcenter, \vdash, \vdots, \vec, \vector,

\vee, \verb, \verbatim, \vert, \Vert, \vfil, \vfill, \vfilneg, \vfootnote, \vfuzz, \vglue, \vline, \voffset, \vphantom, \vrule, \vsize, \vskip, \vspace, \vsplit, \vss, \vtop, \wd, \wedge, \widehat, \widetilde, \widowpenalty, \width, \wlog, \wp, \wr, \write, \xappto, \xdef, \xdefinecolor, \XeLaTeX, \XeTeX, \xi, \Xi, \xleaders, \xpreto, \xspaceskip, \year, \zeta

C. List of Known LaTeX Environments

Below all *predefined* environment names are listed that are treated as "silent" names by CNLTX, that is, those defined by CNLTX-LISTINGS.

array, center, description, displaymath, document, enumerate, eqnarray, equation, figure, flushleft, flushright, itemize, labeling, list, longtable, lrbox, math, minipage, otherlanguage, picture, quote, quoting, sloppypar, tabbing, table, tabu, tabular, tabularx, tabulary, trivlist, verbatim

D. List of Entries in `cnltx.bib`

Most entries in `cnltx.bib` are entries of the `@package` type. The cite keys that the file currently contains are listed below. This list is very likely to be extended significantly in the future.

pkg:abbrevs (@package), pkg:accsupp (@package), pkg:acro (@package), pkg:acromake (@package), pkg:acronym (@package), pkg:acroterm (@package), pkg:adjustbox (@package), pkg:amsmath (@package), pkg:amstext (@package), pkg:answers (@package), pkg:array (@package), pkg:asymptote (@package), pkg:auto-pst-pdf (@package), pkg:babel (@package), pkg:bm (@package), pkg:biblatex (@package), pkg:bigfoot (@package), pkg:bohr (@package), pkg:booktabs (@package), pkg:bpchem (@package), pkg:catchfile (@package), pkg:chemcompounds (@package), pkg:chemcono (@package), pkg:chemfig (@package), pkg:chemformula (@package), pkg:chemgreek (@package), pkg:chemmacros (@package), pkg:chemnum (@package), pkg:chemstyle (@package), pkg:chngcntr (@package), bnd:cnltx (@bundle), cls:cnpkgdoc (@class), pkg:cntformats (@package), pkg:cprotect (@package), pkg:elements (@package), pkg:endnotes (@package), pkg:enotez (@package), pkg:enumitem (@package), pkg:environ (@package), pkg:epic (@package), pkg:eqexam (@package), pkg:esami (@package), pkg:etoolbox (@package), cls:exam (@class), pkg:examdesign (@package), pkg:exercise (@package), pkg:exsheets (@package), pkg:exsol (@package), pkg:fixfoot (@package), pkg:fnpct (@package), pkg:fontenc (@package), pkg:fontspec (@package), pkg:footmisc (@package), pkg:footnote (@package), pkg:fourier (@package), pkg:geometry (@package), pkg:ghsystem (@package), pkg:glossaries (@package), pkg:graphicx (@package), bnd:greek-fontenc (@bundle), pkg:hologo (@package), pkg:hyperref (@package), pkg:idxcmds (@package), pkg:ifluatex (@package), pkg:ifpdf (@package), pkg:ifplatform (@package), pkg:ifxetex (@package), pkg:imakeidx (@package), pkg:inputenc (@package), bnd:koma-script (@bundle), pkg:kpfonts (@package), bnd:l3experimental (@bundle), bnd:l3kernel (@bundle), bnd:l3packages (@bundle), pkg:libertine (@package), pkg:listings (@package), pkg:longtable (@package), pkg:lscape (@package),

pkg:ltxcmds (@package),
pkg:manyfoot (@package),
pkg:marginnote (@package),
pkg:mathdesign (@package),
pkg:mathtools (@package),
pkg:mdframed (@package),
cls:memoir (@class),
pkg:mfirstuc (@package),
pkg:mhchem (@package),
pkg:microtype (@package),
pkg:multicol (@package),
pkg:multienum (@package),
pkg:musixtex (@package),
pkg:newtx (@package),
pkg:nicefrac (@package),
pkg:nomencl (@package),
pkg:parnotes (@package),
pkg:pagenote (@package),
pkg:pdfcomment (@package),
pkg:pdftexcmds (@package),
pkg:perpage (@package),
pkg:pgf (@package),
pkg:pgfopts (@package),
pkg:pgfplots (@package),
pkg:polyglossia (@package),
pkg:probsoln (@package),
pkg:psfrag (@package),
pkg:pstricks (@package),
pkg:pxgreeks (@package),
pkg:ragged2e (@package),
pkg:realbookchords (@package),
pkg:relsize (@package),
pkg:scrlfile (@package),
pkg:sepfootnotes (@package),
pkg:sidenotes (@package),
pkg:siunitx (@package),
pkg:snotez (@package),
pkg:songbook (@package),
pkg:songs (@package),
pkg:splitidx (@package),
pkg:superiors (@package),
pkg:tablefootnote (@package),
pkg:tabu (@package),
pkg:tasks (@package),
pkg:textcomp (@package),
pkg:textgreek (@package),
pkg:tkz-base (@package),
pkg:tkz-euclide (@package),
pkg:tkz-fct (@package),
pkg:translations (@package),
pkg:translator (@package),
pkg:trimspaces (@package),
cls:tufte (@class),
pkg:ulem (@package),
pkg:upgreek (@package),
pkg:xcolor (@package),
pkg:xfrac (@package),
pkg:xpatch (@package),
pkg:xspace (@package),
pkg:xymtex (@package),
bnd:yafoot (@bundle)

E. Bibliography

[Ars11] Donald Arseneau. ulem. Mar. 18, 2011 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/ulem/.

[DS13] Marco Daniel and Elke Schubert.
mdframed. version 1.9b, July 1, 2013 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/mdframed/.

[Gre16] Enrico Gregorio. imakeidx. version 1.3e, Oct. 15, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/imakeidx/.

[HM19] Carsten Heinz and Brooks Moses, current maintainer: Jobst Hoffmann.
listings. version 1.8c, Sept. 10, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/listings/.

[Ker16] Uwe Kern. xcolor. version 2.12, May 11, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/xcolor/.

[Koh18] Markus Kohm. marginnote. version 1.4b, Aug. 9, 2018 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/marginnote/.

[Koh19] Markus Kohm. KOMA-Script. version 3.25, Jan. 14, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/koma-script/.

[Leh19a] Philipp Lehman, current maintainers: Joseph Wright, Audrey Boruvka, and Philip Kime. biblatex. version 3.13, Aug. 7, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/biblatex/.

[Leh19b] Philipp Lehman, current maintainer: Joseph Wright.
etoolbox. version 2.5h, Sept. 21, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/etoolbox/.

[Mit19] Frank Mittelbach. multicol. version 1.8w, Mar. 1, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/required/tools/multicol/.

[Nie13] Clemens Niederberger. cnpkgdoc. version 0.3a, July 13, 2013 (or newer).
URL: https://bitbucket.org/cgnieder/cnpkgdoc/.

[Nie15] Clemens Niederberger. idxcmds. version 0.2c, Aug. 13, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/idxcmds/.

[Nie17] Clemens Niederberger. translations. version 1.7a, Aug. 31, 2017 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Nie19] Clemens Niederberger. acro. version 2.10, Sept. 23, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/acro/.

[Obe16a] Heiko Oberdiek. catchfile. version 1.7, May 16, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/oberdiek/.

[Obe16b] Heiko Oberdiek. ltxcmds. version 1.23, May 16, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/oberdiek/.

[Obe18] Heiko Oberdiek. accsupp. version 0.5, Mar. 28, 2018 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/oberdiek/.

[Obe19] Heiko Oberdiek. pdftexcmds. version 0.30, July 25, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/oberdiek/.

[OR19] Heiko Oberdiek and Sebastian Rahtz.
hyperref. version 6.88i, Sept. 12, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/hyperref/.

[Rah16] Sebastian Rahtz. textcomp. Feb. 18, 2016 (or newer).
URL: http://mirrors.ctan.org/obsolete/fonts/psfonts/ts1.

[Rob09] Will Robertson. trimspaces. version 1.1, Sept. 17, 2009 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/trimspaces/.

[Rob10] Will Robertson. ifxetex. version 0.6, Sept. 12, 2010 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/ifxetex/.

[Sch19a] Martin Scharrer. adjustbox. version 1.2, Jan. 4, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/adjustbox/.

[Sch19b] Martin Schröder. ragged2e. version 2.2, July 28, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/ms/ragged2e/.

[Tal19] Nicola L. C. Talbot. glossaries. version 4.42, Jan. 6, 2019 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/glossaries/.

[Wil18] Peter Wilson, current maintainer: Will Robertson.
chngcntr. version 1.1a, Apr. 9, 2018 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/chngcntr/.

[Wri14] Joseph Wright. pgfopts. version 2.1a, July 10, 2014 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/pgfopts/.

F. Index

D

E

F