The ltxcmds package

Heiko Oberdiek
<heiko.oberdiek at googlemail.com>

2011/03/16 v1.18

Abstract

The package ltxcmds exports some utility macros from the LaTeX kernel into a separate namespace and also provides them for other formats such as plain-TeX.

Contents

1 Documentation

1.1 Introduction

Many of my packages also support other formats such as plain-TeX. Because I am rather familiar with the utility macros from LaTeX's kernel (e.g. `\@gobble`,

\@firstoftwo), I found myself rewriting them again and again, because they are lacking in plain-TEX.

Therefore this package provides often used macros and similar ones with the name prefix \ltx@. This avoids also faulty redefinitions. I remember an example where a package redefined \@firstoftwo with forgetting \long.

1.2 Numbers

\ltx@zero	→	0
\ltx@one	→	1
\ltx@two	→	2
\ltx@cclv	→	255
\ltx@minusone	→	-1

These commands are numbers 0, 1, 2, 255 and -1. They are not digits and a space is not gobbled afterwards. Macro \ltx@minusone is available since version 2010/12/12 v1.15.

1.3 Argument killers

\ltx@gobble {⟨*1*⟩}	→
\ltx@gobbletwo {⟨*1*⟩} {⟨*2*⟩}	→
\ltx@gobblethree {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩}	→
\ltx@gobblefour {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩}	→

\ltx@GobbleNum {⟨*num*⟩} {⟨*1*⟩} {⟨*2*⟩} ... {⟨⟨*num*⟩⟩}	→

The first argument ⟨*num*⟩ of macro \ltx@GobbleNum specifies, how many following arguments are eaten. Macro \ltx@GobbleNum is expandable in exact two expansion steps.

1.4 Argument grabbers

\ltx@firstofone {⟨*1*⟩}	→	⟨*1*⟩
\ltx@firstoftwo {⟨*1*⟩} {⟨*2*⟩}	→	⟨*1*⟩
\ltx@secondoftwo {⟨*1*⟩} {⟨*2*⟩}	→	⟨*2*⟩
\ltx@firstofthree {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩}	→	⟨*1*⟩
\ltx@secondofthree {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩}	→	⟨*2*⟩
\ltx@thirdofthree {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩}	→	⟨*3*⟩
\ltx@firstoffour {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩}	→	⟨*1*⟩
\ltx@secondoffour {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩}	→	⟨*2*⟩
\ltx@thirdoffour {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩}	→	⟨*3*⟩
\ltx@fourthoffour {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩}	→	⟨*4*⟩

Macros \ltx@firstofthree, \ltx@secondofthree and \ltx@thirdofthree were added in version 2010/11/12 v1.11. Macros \ltx@firstoffour, ..., \ltx@forthoffour were added in version 2011/02/04 v1.16.

1.5 List helpers

`\ltx@carzero` ... `\@nil`	→	
`\ltx@cdrzero` ... `\@nil`	→	...

`\ltx@car` {⟨*1*⟩} ... `\@nil`	→	⟨*1*⟩
`\ltx@cdr` {⟨*1*⟩} ... `\@nil`	→	...

`\ltx@cartwo` {⟨*1*⟩} {⟨*2*⟩} ... `\@nil`	→	⟨*1*⟩⟨*2*⟩
`\ltx@cdrtwo` {⟨*1*⟩} {⟨*2*⟩} ... `\@nil`	→	...

`\ltx@carthree` {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} ... `\@nil`	→	⟨*1*⟩⟨*2*⟩⟨*3*⟩
`\ltx@cdrthree` {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} ... `\@nil`	→	...

`\ltx@carfour` {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩} ... `\@nil`	→	⟨*1*⟩⟨*2*⟩⟨*3*⟩⟨*4*⟩
`\ltx@cdrfour` {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩} ... `\@nil`	→	...

`\ltx@CarNum` {⟨*num*⟩} {⟨*1*⟩} ... {⟨⟨*num*⟩⟩} {⟨⟨*num*⟩+*1*⟩} ... `\@nil`	→	{⟨*1*⟩} ... {⟨⟨*num*⟩⟩} ...
`\ltx@CdrNum` {⟨*num*⟩} {⟨*1*⟩} ... {⟨⟨*num*⟩⟩} {⟨⟨*num*⟩+*1*⟩} ... `\@nil`	→	{⟨⟨*num*⟩+*1*⟩} ...

Macros `\ltx@CarNum` and `\ltx@CdrNum` are expandable in exact two expansion steps.

1.6 Tail recursion

`\ltx@ReturnAfterFi` {⟨*1*⟩} `\fi`	→	`\fi` ⟨*1*⟩
`\ltx@ReturnAfterElseFi` {⟨*1*⟩} `\else` {⟨*2*⟩} `\fi`	→	`\fi` ⟨*1*⟩

1.7 Empty macro

`\ltx@empty`	→	

1.8 Characters

`\ltx@space`	→	␣	
`\ltx@percentchar`	→	`%`	
`\ltx@backslashchar`	→	`\`	
`\ltx@hashchar`	→	`#`	(since v1.7)
`\ltx@leftbracechar`	→	{	(since v1.8)
`\ltx@rightbracechar`	→	}	(since v1.8)

1.9 Boolean switch

`\ltx@newif` {⟨*cmd*⟩}

`\ltx@newif` defines a new boolean switch ⟨*cmd*⟩ like `\newif`. Unlike plain TEX's `\newif`, `\ltx@newif` is not `\outer`. The command ⟨*cmd*⟩ must start with the two characters `if`.

`\ltx@newglobalif` {⟨*cmd*⟩}

`\ltx@newglobalif` defines a new boolean switch ⟨*cmd*⟩ like `\ltx@newif`. However the switch setting commands, ⟨*cmd*⟩ without the prefix `if` and followed by `true` or `false` are acting globally.

1.10 Command definitions

`\ltx@ifundefined` {⟨*cmd*⟩} {⟨*yes*⟩} {⟨*no*⟩}

If ε-TEX is available, `\ifcsname` is used that does not have the side effect of defining undefined commands with meaning of `\relax`. This command is always expandable. Change in version 1.1: Also the meaning `\relax` is always considered "undefined".

`\ltx@IfUndefined` {⟨*cmd*⟩} {⟨*yes*⟩} {⟨*no*⟩}

If ε-TEX is available, `\ifcsname` is used that does not have the side effect of defining undefined commands with meaning of `\relax`. Also it always checks for the meaning of `\relax` and considers this as undefined. This macro is not expandable without ε-TEX.

`\ltx@LocalExpandAfter`

It expands the token after the next token but in a local context. That is the difference to `\expandafter`. The local context discards the side effect of `\csname` and let the command undefined after the expansion step.

1.11 Stripping

`\ltx@RemovePrefix`
`\ltx@StripPrefix`

All tokens up to and including the next available character '>' are thrown away. Usually it is used to strip the first part of the output of the commands `\meaning` or `\pdflastmatch`. Macro `\ltx@RemovePrefix` has the same meaning as LATEX's `\strip@prefix`, whereas macro `\ltx@StripPrefix` expands the next token once before stripping the prefix.

`\ltx@onelevel@sanitize` {⟨*macro*⟩}

Macro `\ltx@onelevel@sanitize` provides LATEX's `\@onelevel@sanitize`. The macro is expanded once and the contents is converted to characters with catcode 12 (other) and space tokens with catcode 10 (space). Then then sanitized contents is stored into the macro again. Since version 1.12.

1.12 File management

All macros in this section are expandable like the counterparts of the LaTeX kernel. Also they can be used after the preamble.

1.12.1 File extensions

`\ltx@clsextension`
`\ltx@pkgextension`

Macros `\ltx@clsextension` and `\ltx@styextension` stores the strings `cls` and `sty`. In opposite to LaTeX's `\@clsextension` and `\@styextension` they can also be used after `\begin{document}`.

1.12.2 Load check

`\ltx@ifclassloaded` {⟨*class*⟩} {⟨*yes*⟩} {⟨*no*⟩}
`\ltx@ifpackageloaded` {⟨*package*⟩} {⟨*yes*⟩} {⟨*no*⟩}

Macros `\ltx@ifclassloaded`/`\ltx@ifpackageloaded` execute ⟨*yes*⟩, if the ⟨*class*⟩ or ⟨*package*⟩ is loaded, otherwise ⟨*no*⟩ is called. Both ⟨*class*⟩ and ⟨*package*⟩ are specified without extension. The macros can also be used after `\begin{document}`.

`\ltx@iffileloaded` {⟨*file*⟩} {⟨*yes*⟩} {⟨*no*⟩}

If LaTeX's `\ProvidesFile` macro was called before using ⟨*file*⟩ as argument, then `\ltx@iffileloaded` calls ⟨*yes*⟩, otherwise ⟨*no*⟩. Therefore it is possible that the ⟨*file*⟩ is loaded, but ⟨*no*⟩ is executed because of a missing `\ProvidesFile`. The LaTeX kernel does not have a counterpart of `\ltx@iffileloaded`.

Note that the file name used in `\ProvidesFile` and `\ltx@iffileloaded` must match. For example, if TeX's default extension `..tex` was given in the first command, then it must also specified in the latter command and vice versa.

1.12.3 Version date check

`\ltx@ifclasslater` {⟨*class*⟩} {⟨*date*⟩} {⟨*yes*⟩} {⟨*no*⟩}
`\ltx@ifpackagelater` {⟨*package*⟩} {⟨*date*⟩} {⟨*yes*⟩} {⟨*no*⟩}
`\ltx@iffilelater` {⟨*file*⟩} {⟨*date*⟩} {⟨*yes*⟩} {⟨*no*⟩}

If a `\ProvidesClass`/`\ProvidesPackage`/`\ProvidesFile` command with exact the same class/package/file was executed before with an optional argument that starts with a LaTeX version date, then this version date is compared with the argument ⟨*date*⟩. If they are equal or if the version date is the later date, then ⟨*yes*⟩ is called. In all other cases ⟨*no*⟩ is executed.

A LaTeX date has the format `YYYY/MM/DD` with `YYYY` as year with four digits, `MM` as month with two digits and `DD` as day with two digits. If pdfTeX's `\pdfmatch` is available, then it is used to detect the version date, to reject invalid date formats and to reject some invalid dates. Dates before 1994/01/01 are always invalid, because version dates are introduced with LaTeX 2_ε in 1994.

1.13 Macro additions

`\ltx@GlobalAppendToMacro` {⟨*cmd*⟩} {⟨*addition*⟩}
`\ltx@LocalAppendToMacro` {⟨*cmd*⟩} {⟨*addition*⟩}

The ⟨*addition*⟩ is appended to the parameterless macro ⟨*cmd*⟩. If ⟨*cmd*⟩ is undefined or has the meaning `\relax`, then it will be initialized as empty macro before. Due to a bug ⟨*addition*⟩ must not contain `\par` before version 2010/10/25 v1.9.

1.14 Macro `\ltx@ifnextchar`

`\ltx@ifnextchar` {⟨*char*⟩} {⟨*yes*⟩} {⟨*no*⟩}

If next character is ⟨*char*⟩ then ⟨*yes*⟩ is called, otherwise ⟨*no*⟩. The character is not removed.

1.15 `\ltx@leavevmode`, `\ltx@mbox`

`\ltx@leavevmode`

Macro `\ltx@leavevmode` calls pdfTEX's `\quitvmode`. Otherwise `\leavevmode` is used and defined if it is necessary.

`\ltx@mbox`

Macro `\ltx@mbox` reimplements `\mbox` with two changes. Instead of `\leavevmode` it uses `\ltx@leavevmode` and stops right after `\hbox`. Especially it does not grab the argument and allows the extended syntax of `\hbox`.

1.16 Expandable test for emptiness

`\ltx@ifempty` {⟨*stuff*⟩} {⟨*yes*⟩} {⟨*no*⟩}

Macro `\ltx@ifempty` checks in exact two expansion steps whether ⟨*stuff*⟩ is empty or contains token. Depending on the result ⟨*yes*⟩ or ⟨*no*⟩ is executed. The token in ⟨*stuff*⟩ may contain `\par` and unmatched conditionals (`\if`, `\else`, `\fi`, ...). Since version 2010/11/12 v1.11.

`\ltx@ifblank` {⟨*stuff*⟩} {⟨*yes*⟩} {⟨*no*⟩}

Macro `\ltx@ifblank` tests in exact two expansion steps if ⟨*stuff*⟩ is empty or contain only blank spaces. In this case argument ⟨*yes*⟩ is called. If ⟨*stuff*⟩ contains other tokens than spaces then ⟨*no*⟩ is executed. Since version 2010/12/04 v1.13.

1.17 Stripping spaces

`\ltx@zapspace` {⟨*stuff*⟩}

Macro `\ltx@zapspace` strips spaces from ⟨*stuff*⟩ that are not hidden inside curly braces. Like LaTeX's `\zap@space` it is expandable. Differences:

- Syntax: `\zap@space` also expects a space token and `\@empty` after ⟨*stuff*⟩.
- Macro `\ltx@zapspace` is expandable in exact two expansion steps.

- Macro `\ltx@zapspace` always retains curly braces.
- Macro `\zap@space` has a bug. It stops stripping spaces after a token group in curly braces if the first two tokens inside the group are equal.
- Macro `\ltx@zapspace` also works with `\par` and conditionals (`\if`, `\else`, `\fi`, ...).

Macro `\ltx@zapspace` is available since version 2010/12/07 v1.14.

1.18 Check for emptiness of boxes

`\ltx@IfBoxEmpty` {⟨*box register number*⟩} {⟨*yes*⟩} {⟨*no*⟩}

Macro `\ltx@IfBoxEmpty` calls ⟨*yes*⟩ if the box exists (`\ifvoid` returns false) and the box does not contain any content. Otherwise if the box is void or contains something, then ⟨*no*⟩ is executed. Thus being empty means that the box exists and is either an `\hbox` or a `\vbox` and may even have dimensions other than 0.0 pt, but the box does not contain anything. Macro `\ltx@IfBoxEmpty` is available since 2010/02/04 v1.16.

`\ltx@IfBoxVoidOrEmpty` {⟨*box register number*⟩} {⟨*yes*⟩} {⟨*no*⟩}

Macro `\ltx@IfBoxVoidOrEmpty` calls ⟨*yes*⟩ if the box is either void or does not contain any content. Otherwise ⟨*no*⟩ is executed. Macro `\ltx@IfBoxVoidOrEmpty` is available since 2010/02/04 v1.16.

2 Implementation

2.1 Identification

```
⟨*package⟩
```

Reload check, especially if the package is not used with LaTeX.

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \catcode44=12 % ,
  \catcode45=12 % -
  \catcode46=12 % .
  \catcode58=12 % :
  \catcode64=11 % @
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\let\expandafter\x\csname ver@ltxcmds.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
```

```
      \x{ltxcmds}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup%
```

Package identification:

```
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode35=6 % #
  \catcode39=12 % '
  \catcode40=12 % (
  \catcode41=12 % )
  \catcode44=12 % ,
  \catcode45=12 % -
  \catcode46=12 % .
  \catcode47=12 % /
  \catcode58=12 % :
  \catcode64=11 % @
  \catcode91=12 % [
  \catcode93=12 % ]
  \catcode123=1 % {
  \catcode125=2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@ltxcmds.sty\endcsname
\ProvidesPackage{ltxcmds}%
  [2011/03/16 v1.18 LaTeX kernel commands for general use (HO)]%
\begingroup\catcode61\catcode48\catcode32=10\relax%
  \catcode13=5 % ^^M
  \endlinechar=13 %
  \catcode123=1 % {
  \catcode125=2 % }
  \catcode64=11 % @
  \def\x{\endgroup
    \expandafter\edef\csname LTXcmds@AtEnd\endcsname{%
      \endlinechar=\the\endlinechar\relax
      \catcode13=\the\catcode13\relax
      \catcode32=\the\catcode32\relax
      \catcode35=\the\catcode35\relax
      \catcode61=\the\catcode61\relax
      \catcode64=\the\catcode64\relax
      \catcode123=\the\catcode123\relax
      \catcode125=\the\catcode125\relax
    }%
  }%
\x\catcode61\catcode48\catcode32=10\relax%
\catcode13=5 % ^^M
```

```
\endlinechar=13 %
\catcode35=6 % #
\catcode64=11 % @
\catcode123=1 % {
\catcode125=2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\LTXcmds@AtEnd{%
    \LTXcmds@AtEnd
    \catcode#1=\the\catcode#1\relax
  }%
  \catcode#1=#2\relax
}
\TMP@EnsureCode{36}{3}% $
\TMP@EnsureCode{38}{4}% &
\TMP@EnsureCode{40}{12}% (
\TMP@EnsureCode{41}{12}% )
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{91}{12}% [
\TMP@EnsureCode{96}{12}% `
\TMP@EnsureCode{93}{12}% ]
\TMP@EnsureCode{94}{12}% ^ (superscript) (!)
\TMP@EnsureCode{124}{12}% |
\edef\LTXcmds@AtEnd{\LTXcmds@AtEnd\noexpand\endinput}
```

2.2 Numbers

\ltx@zero

```
\chardef\ltx@zero=0 %
```

\ltx@one

```
\chardef\ltx@one=1 %
```

\ltx@two

```
\chardef\ltx@two=2 %
```

\ltx@active

```
\chardef\ltx@active=13 %
```

\ltx@cclv

```
\chardef\ltx@cclv=255 %
```

\ltx@minusone

```
\def\ltx@minusone{%
  -\ltx@one
}
```

2.3 Argument killers

\ltx@gobble

```
\long\def\ltx@gobble#1{}
```

\ltx@gobbletwo

```
\long\def\ltx@gobbletwo#1#2{}
```

\ltx@gobblethree

```
\long\def\ltx@gobblethree#1#2#3{}
```

\ltx@gobblefour

```
\long\def\ltx@gobblefour#1#2#3#4{}
```

\ltx@GobbleNum

```
\def\ltx@GobbleNum#1{%
  \romannumeral
  \csname ltx@zero%
  \expandafter\LTXcmds@GobbleNum
  \romannumeral\LTXcmds@num{#1}000{m\endcsname}%
}
```

\LTXcmds@GobbleNum

```
\def\LTXcmds@GobbleNum#1{%
  \csname LTXcmds@G#1\LTXcmds@GobbleNum
}
```

\LTXcmds@Gm

```
\long\def\LTXcmds@Gm#1{%
  \endcsname
}
```

2.4 Argument grabbers

\ltx@firstofone

```
\long\def\ltx@firstofone#1{#1}
```

\ltx@firstoftwo

```
\long\def\ltx@firstoftwo#1#2{#1}
```

\ltx@secondoftwo

```
\long\def\ltx@secondoftwo#1#2{#2}
```

\ltx@firstofthree

```
\long\def\ltx@firstofthree#1#2#3{#1}
```

\ltx@secondofthree

```
\long\def\ltx@secondofthree#1#2#3{#2}
```

\ltx@thirdofthree

```
\long\def\ltx@thirdofthree#1#2#3{#3}%
```

\ltx@firstoffour

```
\long\def\ltx@firstoffour#1#2#3#4{#1}
```

\ltx@secondoffour

```
\long\def\ltx@secondoffour#1#2#3#4{#2}
```

\ltx@thirdoffour

```
\long\def\ltx@thirdoffour#1#2#3#4{#3}%
```

\ltx@fourthoffour

```
\long\def\ltx@fourthoffour#1#2#3#4{#4}%
```

2.5 List helpers

\ltx@car

```
\long\def\ltx@car#1#2\@nil{#1}
```

\ltx@cdr

```
\long\def\ltx@cdr#1#2\@nil{#2}
```

\ltx@carzero

```
\long\def\ltx@carzero#1\@nil{}%
```

\ltx@cdrzero

```
\long\def\ltx@cdrzero#1\@nil{#1}%
```

\ltx@cartwo

```
\long\def\ltx@cartwo#1#2#3\@nil{#1#2}
```

\ltx@cdrtwo

```
\long\def\ltx@cdrtwo#1#2#3\@nil{#3}
```

\ltx@carthree

```
\long\def\ltx@carthree#1#2#3#4\@nil{#1#2#3}
```

\ltx@cdrthree

```
\long\def\ltx@cdrthree#1#2#3#4\@nil{#4}
```

\ltx@carfour

```
\long\def\ltx@carfour#1#2#3#4#5\@nil{#1#2#3#4}
```

\ltx@cdrfour

```
\long\def\ltx@cdrfour#1#2#3#4#5\@nil{#5}
```

\ltx@CarNum

```
\def\ltx@CarNum#1{%
  \romannumeral
  \csname LTXcmds@CarNumFinish%
  \expandafter\LTXcmds@CarNum
  \romannumeral\LTXcmds@num{#1}000{x\endcsname}%
}
```

\LTXcmds@CarNum

```
\def\LTXcmds@CarNum#1{%
  \csname LTXcmds@C#1\LTXcmds@CarNum
}
```

\LTXcmds@Cm

```
\long\def\LTXcmds@Cm#1#2{%
  \endcsname{#1#2}%
}
```

\LTXcmds@Cx

```
\def\LTXcmds@Cx#1{%
  \endcsname{}%
}
```

\LTXcmds@CarNumFinish

```
\long\def\LTXcmds@CarNumFinish#1#2\@nil{%
  \ltx@zero
  #1%
}
```

\ltx@CdrNum

```
\def\ltx@CdrNum#1{%
  \romannumeral0%
  \expandafter\expandafter\expandafter\LTXcmds@CdrNum
  \ltx@GobbleNum{#1}%
}
```

\LTXcmds@CdrNum

```
\long\def\LTXcmds@CdrNum#1\@nil{ #1}%
```

2.6 Tail recursion

\ltx@ReturnAfterFi

```
\long\def\ltx@ReturnAfterFi#1\fi{\fi#1}
```

\ltx@ReturnAfterElseFi

```
\long\def\ltx@ReturnAfterElseFi#1\else#2\fi{\fi#1}
```

2.7 Empty macro

\ltx@empty

```
\def\ltx@empty{}
```

2.8 Characters

\ltx@space

```
\def\ltx@space{ }
```

\ltx@percentchar

```
\begingroup
  \lccode`0=`\%\relax
\lowercase{\endgroup
  \def\ltx@percentchar{0}%
}
```

\ltx@backslashchar

```
\begingroup
  \lccode`0=`\\\relax
\lowercase{\endgroup
  \def\ltx@backslashchar{0}%
}
```

\ltx@hashchar

```
\begingroup
  \lccode`0=`\#\relax
\lowercase{\endgroup
  \def\ltx@hashchar{0}%
}
```

\ltx@leftbracechar

```
\begingroup
  \lccode`0=`\{\relax
\lowercase{\endgroup
  \def\ltx@leftbracechar{0}%
}
```

\ltx@rightbracechar

```
\begingroup
  \lccode`0=`\}\relax
\lowercase{\endgroup
  \def\ltx@rightbracechar{0}%
}
```

2.9 Boolean switch

\ltx@newif

```
\def\ltx@newif#1{%
  \begingroup
    \escapechar=-1 %
  \expandafter\endgroup
  \expandafter\LTXcmds@newif\string#1\@nil
}
```

\LTXcmds@newif

```
\begingroup
  \escapechar=-1 %
\expandafter\endgroup
\expandafter\def\expandafter\LTXcmds@newif\string\if#1\@nil{%
  \expandafter\edef\csname#1true\endcsname{%
    \let
    \expandafter\noexpand\csname if#1\endcsname
    \noexpand\iftrue
  }%
  \expandafter\edef\csname#1false\endcsname{%
    \let
    \expandafter\noexpand\csname if#1\endcsname
    \noexpand\iffalse
  }%
  \csname#1false\endcsname
}
```

\ltx@newglobalif

```
\def\ltx@newglobalif#1{%
  \begingroup
    \escapechar=-1 %
  \expandafter\endgroup
  \expandafter\LTXcmds@newglobalif\string#1\@nil
}
```

\LTXcmds@newglobalif

```
\begingroup
  \escapechar=-1 %
\expandafter\endgroup
\expandafter
\def\expandafter\LTXcmds@newglobalif\string\if#1\@nil{%
  \expandafter\edef\csname#1true\endcsname{%
    \global\let
    \expandafter\noexpand\csname if#1\endcsname
    \noexpand\iftrue
  }%
  \expandafter\edef\csname#1false\endcsname{%
    \global\let
    \expandafter\noexpand\csname if#1\endcsname
    \noexpand\iffalse
  }%
  \csname#1false\endcsname
}
```

2.10 Command definitions

\ltx@LocalExpandAfter

```
\def\ltx@LocalExpandAfter{%
  \begingroup
    \expandafter\expandafter\expandafter
  \endgroup
  \expandafter
}

\ltx@LocalExpandAfter
\ifx\csname ifcsname\endcsname\relax
```

\ltx@ifundefined

```
  \def\ltx@ifundefined#1{%
    \expandafter\ifx\csname #1\endcsname\relax
      \expandafter\ltx@firstoftwo
    \else
      \expandafter\ltx@secondoftwo
    \fi
  }%
```

\ltx@IfUndefined

```
  \def\ltx@IfUndefined#1{%
    \begingroup\expandafter\expandafter\expandafter\endgroup
    \expandafter\ifx\csname #1\endcsname\relax
      \expandafter\ltx@firstoftwo
    \else
      \expandafter\ltx@secondoftwo
    \fi
  }%

  \expandafter\ltx@gobble
\else
  \expandafter\ltx@firstofone
\fi
{%
```

\ltx@ifundefined

```
  \def\ltx@ifundefined#1{%
    \ifcsname #1\endcsname
      \expandafter\ifx\csname #1\endcsname\relax
        \expandafter\expandafter\expandafter\ltx@firstoftwo
      \else
        \expandafter\expandafter\expandafter\ltx@secondoftwo
      \fi
    \else
      \expandafter\ltx@firstoftwo
    \fi
  }%
```

\ltx@IfUndefined

```
  \let\ltx@IfUndefined\ltx@ifundefined

}
```

2.11 Stripping

\ltx@RemovePrefix

```
\def\ltx@RemovePrefix#1>{}
```

\ltx@StripPrefix

```
\def\ltx@StripPrefix{%
  \expandafter\ltx@RemovePrefix
}
```

\ltx@onelevel@sanitize

```
\def\ltx@onelevel@sanitize#1{%
  \edef#1{%
    \expandafter
    \ltx@RemovePrefix\meaning#1%
  }%
}
```

2.12 File management

2.12.1 File extensions

\ltx@clsextension

```
\def\ltx@clsextension{cls}
```

\ltx@pkgextension

```
\def\ltx@pkgextension{sty}
```

2.12.2 Load check

\ltx@iffileloaded

```
\def\ltx@iffileloaded#1{%
  \ltx@ifundefined{ver@#1}\ltx@secondoftwo\ltx@firstoftwo
}
```

\ltx@ifclassloaded

```
\def\ltx@ifclassloaded#1{%
  \ltx@iffileloaded{#1.\ltx@clsextension}%
}
```

\ltx@ifpackageloaded

```
\def\ltx@ifpackageloaded#1{%
  \ltx@iffileloaded{#1.\ltx@pkgextension}%
}
```

2.12.3 Version date check

\ltx@iffilelater

```
\def\ltx@iffilelater#1#2{%
  \ltx@iffileloaded{#1}{%
    \expandafter\LTXcmds@IfLater\expandafter{%
      \number
      \expandafter\expandafter\expandafter\LTXcmds@ParseVersion
      \expandafter\expandafter\expandafter{%
        \csname ver@#1\endcsname
      }%
    \expandafter}\expandafter{%
      \number
      \expandafter\LTXcmds@ParseVersion\expandafter{#2}%
    }%
  }\ltx@secondoftwo
}
```

```
\LTXcmds@IfLater
\def\LTXcmds@IfLater#1#2{%
  \ifcase 0%
      \ifnum#1<19940101 %
      \else
        \ifnum#2<19940101 %
        \else
          \ifnum#2>#1 %
          \else
            1%
          \fi
        \fi
      \fi
      \ltx@space
    \expandafter\ltx@secondoftwo
  \else
    \expandafter\ltx@firstoftwo
  \fi
}
\ltx@ifclasslater
\def\ltx@ifclasslater#1{%
  \ltx@iffilelater{#1.\ltx@clsextension}%
}
\ltx@ifpackagelater
\def\ltx@ifpackagelater#1{%
  \ltx@iffilelater{#1.\ltx@pkgextension}%
}
\ltx@IfUndefined{pdfmatch}{%
\LTXcmds@ParseVersion
  \def\LTXcmds@ParseVersion#1{%
    \LTXcmds@@ParseVersion#10000/00/00\@nil
  }%
\LTXcmds@@ParseVersion
  \def\LTXcmds@@ParseVersion#1#2#3#4/#5#6/#7#8#9\@nil{%
    #1#2#3#4#5#6#7#8%
  }%
}{%
\LTXcmds@ParseVersion
  \def\LTXcmds@ParseVersion#1{%
    \ifnum\pdfmatch{%
      ^%
      (199[4-9]|[2-9][0-9][0-9][0-9])/%
      (0[1-9]|1[0-2])/%
      (0[1-9]|[1-2][0-9]|3[0-1])%
    }{#1}=1 %
      \ltx@StripPrefix\pdflastmatch1 %
      \ltx@StripPrefix\pdflastmatch2 %
      \ltx@StripPrefix\pdflastmatch3 %
    \else
      0%
    \fi
  }%
}
```

2.13 Macro additions

\ltx@GlobalAppendToMacro

```
\long\def\ltx@GlobalAppendToMacro#1#2{%
  \ifx\ltx@undefined#1%
    \let#1\ltx@empty
  \else
    \ifx\relax#1%
      \let#1\ltx@empty
    \fi
  \fi
  \begingroup
    \toks0\expandafter{#1#2}%
    \xdef#1{\the\toks0}%
  \endgroup
}
```

\ltx@LocalAppendToMacro

```
\long\def\ltx@LocalAppendToMacro#1#2{%
  \global\let\LTXcmds@gtemp#1%
  \ifx\ltx@undefined\LTXcmds@gtemp
    \global\let\LTXcmds@gtemp\ltx@empty
  \else
    \ifx\relax\LTXcmds@gtemp
      \global\letLTXcmds@gtemp\ltx@empty
    \fi
  \fi
  \begingroup
    \toks0\expandafter{\LTXcmds@gtemp#2}%
    \xdef\LTXcmds@gtemp{\the\toks0}%
  \endgroup
  \let#1\LTXcmds@gtemp
}
```

2.14 Macro \ltx@ifnextchar

\ltx@ifnextchar

```
\long\def\ltx@ifnextchar#1#2#3{%
  \begingroup
  \let\LTXcmds@CharToken= #1\relax
  \toks\ltx@zero{#2}%
  \toks\ltx@two{#3}%
  \futurelet\LTXcmds@LetToken\LTXcmds@ifnextchar
}
```

\LTXcmds@ifnextchar

```
\def\LTXcmds@ifnextchar{%
  \ifx\LTXcmds@LetToken\LTXcmds@CharToken
    \expandafter\endgroup\the\toks\expandafter\ltx@zero
  \else
    \ifx\LTXcmds@LetToken\LTXcmds@SpaceToken
      \expandafter\expandafter\expandafter\LTXcmds@@ifnextchar
    \else
      \expandafter\endgroup\the\toks
      \expandafter\expandafter\expandafter\ltx@two
    \fi
  \fi
}
```

\LTXcmds@@ifnextchar

```
\begingroup
```

```
  \def\x#1{\endgroup
    \def\LTXcmds@@ifnextchar#1{%
      \futurelet\LTXcmds@LetToken\LTXcmds@ifnextchar
    }%
  }%
\x{ }
```

\LTXcmds@SpaceToken

```
\begingroup
  \def\x#1{\endgroup
    \let\LTXcmds@SpaceToken= #1%
  }%
\x{ }
```

2.15 \ltx@leavevmode, \ltx@mbox

\ltx@leavevmode

```
\ltx@IfUndefined{quitvmode}{%
  \ltx@IfUndefined{leavevmode}{%
    \ltx@IfUndefined{voidb@x}{%
      \ltx@IfUndefined{newbox}{%
        \def\ltx@leavevmode{%
          \begingroup
            \setbox\ltx@zero=\hbox{}%
            \begingroup
              \setbox\ltx@zero=\hbox{\box\ltx@zero}%
            \endgroup
            \unhbox\ltx@zero
          \endgroup
        }%
      }{%
        \csname newbox\endcsname\LTXcmds@VoidBox
        \ifvoid\LTXcmds@VoidBox
        \else
          \setbox\LTXcmds@VoidBox=\hbox{}%
          \begingroup
            \setbox\LTXcmds@VoidBox=\hbox{\box\LTXcmds@VoidBox}%
          \endgroup
        \fi
        \def\ltx@leavevmode{\unhbox\LTXcmds@VoidBox}%
      }%
    }{%
      \def\ltx@leavevmode{\unhbox\voidb@x}%
    }%
  }{%
    \let\ltx@leavevmode\leavevmode
  }%
}{%
  \let\ltx@leavevmode\quitvmode
}
```

\ltx@mbox

```
\def\ltx@mbox{%
  \ltx@leavevmode
  \hbox
}
```

2.16 Help macros

\LTXcmds@num

```
\ltx@IfUndefined{numexpr}{%
```

```
  \def\LTXcmds@num#1{%
    \expandafter\ltx@firstofone\expandafter{%
      \number#1%
    }%
  }%
}{%
  \def\LTXcmds@num#1{%
    \expandafter\ltx@firstofone\expandafter{%
      \the\numexpr#1%
    }%
  }%
}
```

2.17 Expandable test for emptiness

```
\ltx@IfUndefined{detokenize}{%
```

2.17.1 Vanilla TEX

`\ltx@ifempty` The macro is based on `\@ifempty` of Robert R. Schneck [1] and `\@ifnull` of Ulrich Diez [2]. There are three cases to consider:

1. `#1` is empty,
2. `#1` is not empty and the first token is not a begingroup character,
3. `#1` starts with a begingroup character (catcode 1).

```
  \def\LTXcmds@temp#1{%
    \long\def\ltx@ifempty##1{%
      \romannumeral0%
      \iffalse{\fi
        \expandafter\ltx@gobble\expandafter{%
          \expandafter{\string##1}%
          \expandafter\ltx@gobble\string
        }%
        \expandafter\ltx@firstofthree\expandafter
        {\iffalse}\fi
        \expandafter#1\ltx@secondoftwo
      }%
      \expandafter#1\ltx@firstoftwo
    }%
```

`\ltx@ifblank`

```
    \long\def\ltx@ifblank##1{%
      \romannumeral0%
      \iffalse{\fi
        \expandafter\expandafter\expandafter\ltx@gobble
        \expandafter\expandafter\expandafter{%
          \expandafter\expandafter\expandafter{%
            \expandafter\string\ltx@gobble##1.%
          }%
          \expandafter\ltx@gobble\string
        }%
        \expandafter\ltx@firstofthree\expandafter
        {\iffalse}\fi
        \expandafter#1\ltx@secondoftwo
      }%
      \expandafter#1\ltx@firstoftwo
    }%
  }%
  \LTXcmds@temp{ }%

}{%
```

2.17.2 With \detokenize

Ahmed Musa provided \ifstrempty using \detokenize and \pdfstrcmp [3]. Ulrich Diez, GL, Heiko Oberdiek improved it further by removing \pdfstrcmp and taking three arguments [4, 5, 6, 7, 8].

\ltx@ifempty

```
\long\def\ltx@ifempty#1{%
  \romannumeral%
  \csname
    LTXcmds@ifempty%
    \ifcat$\detokenize{#1}$%
      @%
    \fi
  \endcsname
}%
```

\LTXcmds@ifempty@

```
\long\def\LTXcmds@ifempty@#1#2{0 #1}%
```

\LTXcmds@ifempty

```
\long\def\LTXcmds@ifempty#1#2{0 #2}%
```

2.17.3 \ltx@ifblank

\ltx@ifblank

```
\long\def\ltx@ifblank#1{%
  \romannumeral%
  \csname
    LTXcmds@ifempty%
    \ifcat$\detokenize\expandafter{\ltx@gobble#1.}$%
      @%
    \fi
  \endcsname
}%

}
```

2.18 \ltx@zapspace

\ltx@zapspace

```
\long\def\ltx@zapspace#1{%
  \romannumeral
  \LTXcmds@zapspace\ltx@zero#1 \@nil
}
```

\LTXcmds@zapspace

```
\long\def\LTXcmds@zapspace#1 #2\@nil{%
  \ltx@ifempty{#2}{%
    #1%
  }{%
    \LTXcmds@zapspace#1#2\@nil
  }%
}
```

2.19 \ltx@IfBoxEmpty

In case of ε-TEX the test for an empty box is done via \lastnodetype as suggested by David Kastrup [9].

```
\ltx@IfUndefined{lastnodetype}{%
```

```
  \catcode`\$=9 %
  \catcode`\&=14 %
}{%
  \catcode`\$=14 %
  \catcode`\&=9 %
}
```

`\ltx@IfBoxEmpty`

```
\def\ltx@IfBoxEmpty#1{%
  \ifvoid#1\relax
    \expandafter\ltx@secondoftwo
  \else
```

Implementation using ε-TEX's `\lastnodetype`.

```
&   \begingroup
&     \setbox\ltx@zero=\ifhbox#1\hbox\else\vbox\fi{%
&       \ifhmode\unhcopy\else\unvcopy\fi#1\relax
&       \expandafter
&     }%
&   \expandafter\endgroup
&   \ifnum\lastnodetype<\ltx@zero
&     \expandafter\expandafter\expandafter\ltx@firstoftwo
&   \else
&     \expandafter\expandafter\expandafter\ltx@secondoftwo
&   \fi
```

Implementation without ε-TEX using a signature at the beginning of the test box.

```
$   \begingroup
$     \setbox\ltx@zero=\ifhbox#1\hbox\else\vbox\fi{%
$       \penalty\ltx@one
$       \ifhmode\unhcopy\else\unvcopy\fi#1\relax
$       \expandafter
$     }%
$     \ifnum\lastpenalty=\ltx@one
```

Box 0 has been changed and is restored by closing the group.

```
$       \endgroup
$       \begingroup
$       \setbox\ltx@zero=\ifhbox#1\hbox\else\vbox\fi{%
$         \penalty\ltx@two
$         \ifhmode\unhcopy\else\unvcopy\fi#1\relax
$         \expandafter
$       }%
$       \ifnum\lastpenalty=\ltx@two
$         \def\next{\endgroup\expandafter\ltx@firstoftwo}%
$       \else
$         \def\next{\endgroup\expandafter\ltx@secondoftwo}%
$       \fi
$     \else
$       \def\next{\endgroup\expandafter\ltx@secondoftwo}%
$     \fi
$   \next
  \fi
}
```

`\ltx@IfBoxVoidOrEmpty`

```
\def\ltx@IfBoxVoidOrEmpty#1{%
  \ifvoid#1\relax
    \expandafter\ltx@thirdoffour
  \fi
  \ltx@IfBoxEmpty{#1}%
}
```

```
\LTXcmds@AtEnd%
```
⟨/package⟩

3 Test

3.1 Catcode checks for loading

⟨*test1⟩

```
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\RangeCatcodeCheck#1#2#3{%
```

```
  \count@=#1\relax
  \loop
    \ifnum#3=\catcode\count@
    \else
      \errmessage{%
        Character \the\count@\space
        with wrong catcode \the\catcode\count@\space
        instead of \number#3%
      }%
    \fi
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\def\space{ }
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input ltxcmds.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\%=14 %
  \LoadCommand
  \RangeCatcodeCheck{0}{36}{15}%
  \RangeCatcodeCheck{37}{37}{14}%
  \RangeCatcodeCheck{38}{47}{15}%
  \RangeCatcodeCheck{48}{57}{12}%
  \RangeCatcodeCheck{58}{63}{15}%
  \RangeCatcodeCheck{64}{64}{12}%
  \RangeCatcodeCheck{65}{90}{11}%
  \RangeCatcodeCheck{91}{91}{15}%
  \RangeCatcodeCheck{92}{92}{0}%
  \RangeCatcodeCheck{93}{96}{15}%
  \RangeCatcodeCheck{97}{122}{11}%
  \RangeCatcodeCheck{123}{255}{15}%
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

3.2 Test `\ltx@GobbleNum`

```
⟨*test-gobble⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input ltxcmds.sty\relax
\else
  \RequirePackage{ltxcmds}[2011/03/16]%
\fi
\catcode`\@=11 %
\def\msg#{\immediate\write16}%
\msg{[Test \string\ltx@GobbleNum]}%
\long\def\Test#1=#2\\{%
  \edef\StrA{\ltx@GobbleNum#1}%
  \expandafter\expandafter\expandafter\def
```

```
  \expandafter\expandafter\expandafter\StrAA
  \expandafter\expandafter\expandafter{\ltx@GobbleNum#1}%
  \edef\StrB{#2}%
  \ifx\StrA\StrB
    \ifx\StrAA\StrB
      \msg{* ok.}%
    \else
      \msg{StrAA: \StrAA}%
      \msg{StrB: \StrB}%
      \errhelp{Test: #1=#2}%
      \errmessage{Test (two expansions) failed}%
    \fi
  \else
    \msg{StrA: \StrA}%
    \msg{StrB: \StrB}%
    \errhelp{Test: #1=#2}%
    \errmessage{Test (edef) failed!}%
  \fi
}
\Test0abc=abc\\
\Test1abc=bc\\
\Test2abc=c\\
\Test3abcd=d\\
\Test4abcde=e\\
\Test5abcdef=f\\
\Test6abcdefg=g\\
\Test7abcdefgh=h\\
\Test8abcdefghi=i\\
\Test9abcdefghij=j\\
\Test{10}0123456789X=X\\
\Test{12}abcdefghijklm=m\\
\Test{700}%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
X=X\\
\Test{-1}abc=abc\\
\Test2\par\par\relax=\relax\\

\begingroup
  \count1=2 %
  \Test{\count1}abc=c\\%
\endgroup

\ltx@IfUndefined{numexpr}{%
}{%
  \Test{1+1}abc=c\\%
}

\msg{[Test \string\ltx@CdrNum]}%
\long\def\Test#1=#2\\{%
  \edef\StrA{\ltx@CdrNum#1\@nil}%
  \expandafter\expandafter\expandafter\def
  \expandafter\expandafter\expandafter\StrAA
  \expandafter\expandafter\expandafter{\ltx@CdrNum#1\@nil}%
```

```
  \edef\StrB{#2}%
  \ifx\StrA\StrB
    \ifx\StrAA\StrB
      \msg{* ok.}%
    \else
      \msg{StrAA: \meaning\StrAA}%
      \msg{StrB: \meaning\StrB}%
      \errhelp{Test: #1=#2}%
      \errmessage{Test (two expansions) failed}%
    \fi
  \else
    \msg{StrA: \StrA}%
    \msg{StrB: \StrB}%
    \errhelp{Test: #1=#2}%
    \errmessage{Test (edef) failed!}%
  \fi
}
\Test0abc=abc\\
\Test1abc=bc\\
\Test2abc=c\\
\Test3abcd=d\\
\Test4abcde=e\\
\Test5abcdef=f\\
\Test6abcdefg=g\\
\Test7abcdefgh=h\\
\Test8abcdefghi=i\\
\Test9abcdefghij=j\\
\Test{10}0123456789X=X\\
\Test{12}abcdefghijklm=m\\
\Test{700}%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
X=X\\
\Test{-1}abc=abc\\
\Test2\par\par\relax=\relax\\

\msg{[Test \string\ltx@CarNum]}%
\long\def\Test#1=#2\\{%
  \edef\StrA{\ltx@CarNum#1\@nil}%
  \expandafter\expandafter\expandafter\def
  \expandafter\expandafter\expandafter\StrAA
  \expandafter\expandafter\expandafter{\ltx@CarNum#1\@nil}%
  \edef\StrB{#2}%
  \ifx\StrA\StrB
    \ifx\StrAA\StrB
      \msg{* ok.}%
    \else
      \msg{StrAA: \meaning\StrAA}%
      \msg{StrB: \meaning\StrB}%
      \errhelp{Test: #1=#2}%
      \errmessage{Test (two expansions) failed}%
    \fi
  \else
    \msg{StrA: \StrA}%
```

```
    \msg{StrB: \StrB}%
    \errhelp{Test: #1=#2}%
    \errmessage{Test (edef) failed!}%
  \fi
}
\Test0abc=\\
\Test1abc=a\\
\Test2abc=ab\\
\Test3abc=abc\\
\Test3abcd=abc\\
\Test4abcde=abcd\\
\Test{10}0123456789X=0123456789\\
\Test{12}abcdefghijklm=abcdefghijkl\\
\Test{700}%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
X=%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
0123456789012345678901234567890123456789012345678901234567890123456789%
\\
\Test{-1}abc=\\
\Test2\par\par\relax=\par\par\\
\csname @@end\endcsname\end
```
⟨/test-gobble⟩

3.3 Test \ltx@ifempty

```
⟨*test-ifempty⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\errorcontextlines=1000 %
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input ltxcmds.sty\relax
\else
  \RequirePackage{ltxcmds}[2011/03/16]%
\fi
\def\msg#{\immediate\write16}
\def\TestY{\Y}
\def\TestN{\N}
\msg{* \string\ltx@ifempty}
\long\def\test#1{%
  \begingroup
    % Calculate expected test result via macro definition
    \def\Stuff{#1}%
```

```
    \ifx\Stuff\ltx@empty
      \def\StuffEmpty{\Y}%
    \else
      \def\StuffEmpty{\N}%
    \fi
    % Test \ltx@ifempty
    \expandafter\expandafter\expandafter\def
    \expandafter\expandafter\expandafter\TestEmpty
    \expandafter\expandafter\expandafter{%
      \ltx@ifempty{#1}{\Y}{\N}%
    }%
    \ifx\StuffEmpty\TestEmpty
      \msg{* Test OK}%
    \else
      \ltx@IfUndefined{detokenize}{}{%
        \msg{Stuff: [\detokenize{\Stuff}]}%
      }%
      \errmessage{Test failed!}%
    \fi
  \endgroup
}
\test{}
\test{a}
\test{abc}
\test{\par}
\test{ }
\test{\if}
\test{{\if}}
\test{\else}
\test{{\else}}
\test{\fi}
\test{{}\fi}
\test{\or\ifcase}
\test{{}}
\test{{a}}
\test{{}abc}
\test{{\par}}
\test{{}\par}
\def\SpaceTwo#1{%
  \def\SpaceTwo{#1#1}%
}\SpaceTwo{ }
\msg{* \string\ltx@ifblank}
\long\def\test#1{%
  \begingroup
    % Calculate expected test result via macro definition
    \def\Stuff{#1}%
    \ifx\Stuff\ltx@empty
      \def\StuffEmpty{\Y}%
    \else
      \ifx\Stuff\ltx@space
        \def\StuffEmpty{\Y}%
      \else
        \ifx\Stuff\SpaceTwo
          \def\StuffEmpty{\Y}%
        \else
          \def\StuffEmpty{\N}%
        \fi
      \fi
    \fi
    % Test \ltx@ifblank
    \expandafter\expandafter\expandafter\def
    \expandafter\expandafter\expandafter\TestEmpty
```

```
    \expandafter\expandafter\expandafter{%
      \ltx@ifblank{#1}{\Y}{\N}%
    }%
    \ifx\StuffEmpty\TestEmpty
      \msg{* Test OK}%
    \else
      \ltx@IfUndefined{detokenize}{}{%
        \msg{Stuff: [\detokenize{\Stuff}]}%
      }%
      \errmessage{Test failed!}%
    \fi
  \endgroup
}
\test{}
\test{a}
\test{\if}
\test{\else}
\test{\fi}
\test{ \fi}
\test{\par}
\test{ \par}
\test{{}}
\test{ {}}
\def\x#1{%
  \test{#1#1}%
  \test{#1#1{}}%
  \test{#1#1\par}%
  \test{#1#1\else}%
}\x{ }
\csname @@end\endcsname\end
```
⟨/test-ifempty⟩

3.4 Test \ltx@zap@space

```
⟨*test-zapspace⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\errorcontextlines=1000 %
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input ltxcmds.sty\relax
\else
  \RequirePackage{ltxcmds}[2011/03/16]%
\fi
\def\msg#{\immediate\write16}
\def\space{ }
\def\empty{}
\msg{* \string\ltx@zapspace}
\long\def\test#1#2{%
  \begingroup
    \def\TestInput{#1}%
    \def\TestExpected{#2}%
    % Test \ltx@zapspace
    \expandafter\expandafter\expandafter\def
    \expandafter\expandafter\expandafter\TestResult
    \expandafter\expandafter\expandafter{%
      \ltx@zapspace{#1}%
    }%
    \ifx\TestResult\TestExpected
      \msg{* Test OK}%
    \else
```

```
      \ltx@onelevel@sanitize\TestInput
      \ltx@onelevel@sanitize\TestExpected
      \ltx@onelevel@sanitize\TestResult
      \msg{* Input: \space\space\space[\TestInput]}%
      \msg{ \space Result: \space\space[\TestResult]}%
      \msg{ \space Expected: [\TestExpected]}%
      \errmessage{Test failed!}%
    \fi
  \endgroup
}
\long\def\etest#1#2{%
  \begingroup
    \edef\x{\endgroup
      \noexpand\test{#1}{#2}%
    }%
  \x
}
\catcode`\~=13 %
\let~\noexpand
\test{}{}
\test{{}}{{}}
\test{ {}}{{}}
\test{{ }}{{ }}
\test{{} }{{}}
\test{ {} }{{}}
\test{ { } }{{ }}
\test{a {b} c}{a{b}c}
\test{a bb ccc}{abbccc}
\test{{a} {bb} {ccc}}{{a}{bb}{ccc}}
\test{\par}{\par}
\test{\if}{\if}
\test{\space}{\space}
\etest{\par\space\par}{\par\par}
\etest{~\empty\space~\empty}{~\empty~\empty}
\etest{~\fi\space~\else\space}{~\fi~\else}
\csname @@end\endcsname\end
```
⟨/test-zapspace⟩

3.5 Test `\ltx@IfBoxEmpty`

```
⟨*test-ifboxempty⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input ltxcmds.sty\relax
\else
  \RequirePackage{ltxcmds}[2011/03/16]%
\fi
\def\msg#{\immediate\write16}
 % make box 0 void
\begingroup
  \setbox0=\box0 %
\endgroup
\ifvoid0 %
\else
  \errmessage{Voiding box 0 failed}%
\fi
\setbox2=\box0 %
\def\test#1#2{%
  \@test{#1}{#2}%
```

```
  \@@test{#1}{#2}%
  \chardef\x=#1%
  \@test\x{#2}%
  \@@test\x{#2}%
}
\def\@test#1#2{%
  \begingroup
    \setbox9=\hbox{%
      \def\TestExpected{#2}%
      \ltx@IfBoxEmpty{#1}{%
        \def\TestResult{Y}%
      }{%
        \def\TestResult{N}%
      }%
      \ifx\TestExpected\TestResult
        \msg{* Test passed.}%
      \else
        \errmessage{Test failed!}%
      \fi
    }%
    \ifdim\wd9=0pt %
    \else
      \errmessage{Unwanted space?}%
    \fi
  \endgroup
}
\def\@@test#1#2{%
  \begingroup
    \setbox9=\hbox{%
      \def\TestExpected{#2}%
      \ifvoid#1\def\TestExpected{Y}\fi
      \ltx@IfBoxVoidOrEmpty{#1}{%
        \def\TestResult{Y}%
      }{%
        \def\TestResult{N}%
      }%
      \ifx\TestExpected\TestResult
        \msg{* Test passed.}%
      \else
        \errmessage{Test failed!}%
      \fi
    }%
    \ifdim\wd9=0pt %
    \else
      \errmessage{Unwanted space?}%
    \fi
  \endgroup
}
\test0N
\test2N
\setbox0=\hbox{}
\test0Y
\setbox2=\hbox{}
\test2Y
\setbox0=\vbox{}
\test0Y
\setbox2=\vbox{}
\test0Y
\setbox0=\hbox{ }%
\test0N
\setbox2=\hbox{ }%
\test2N
```

```
\setbox0=\hbox{\penalty1}%
\test0N
\setbox2=\hbox{\penalty1}%
\test2N
\csname @@end\endcsname\end
⟨/test-ifboxempty⟩
```

4 Installation

4.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/ltxcmds.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/ltxcmds.pdf Documentation.

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package attachfile2 comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting docstrip archive. The files are extracted by running the `.dtx` through plain TEX:

```
tex ltxcmds.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
ltxcmds.sty                    → tex/generic/oberdiek/ltxcmds.sty
ltxcmds.pdf                    → doc/latex/oberdiek/ltxcmds.pdf
test/ltxcmds-test1.tex         → doc/latex/oberdiek/test/ltxcmds-test1.tex
test/ltxcmds-test-gobble.tex   → doc/latex/oberdiek/test/ltxcmds-test-gobble.tex
test/ltxcmds-test-ifempty.tex  → doc/latex/oberdiek/test/ltxcmds-test-ifempty.tex
test/ltxcmds-test-zapspace.tex → doc/latex/oberdiek/test/ltxcmds-test-zapspace.tex
test/ltxcmds-test-ifboxempty.tex → doc/latex/oberdiek/test/ltxcmds-test-ifboxempty.tex
ltxcmds.dtx                    → source/latex/oberdiek/ltxcmds.dtx
```

If you have a `docstrip.cfg` that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

[1] ftp://ftp.ctan.org/tex-archive/

4.4 Refresh file name databases

If your TEX distribution (teTEX, mikTEX, ...) relies on file name databases, you must refresh these. For example, teTEX users run `texhash` or `mktexlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is pdftk, e.g. unpack the file into the current directory:

```
pdftk ltxcmds.pdf unpack_files output .
```

Unpacking with LATEX. The `.dtx` chooses its action depending on the format:

plain TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

If you insist on using LATEX for docstrip (really, docstrip does not need LATEX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{ltxcmds.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLATEX:

```
pdflatex ltxcmds.dtx
makeindex -s gind.ist ltxcmds.idx
pdflatex ltxcmds.dtx
makeindex -s gind.ist ltxcmds.idx
pdflatex ltxcmds.dtx
```

5 References

[1] Robert R. Schneck: *Re: `\ifempty` solution (was Macro puzzle: maximally general `\ifempty`)*; newsgroup comp.text.tex, news:3eef1ada_6@corp.newsgroups.com, 2003-06-17. http://groups.google.com/group/comp.text.tex/msg/be03a159ec374895

[2] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:ibk3t8$ee7$1@news.albasani.net, 2010-11-12. http://groups.google.com/group/comp.text.tex/msg/803bd57221a04996

[3] Ahmed Musa: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:f5496afe-40ed-42bd-b629-a2419ecf7c0d@o14g2000prn.googlegroups.com, 2010-12-03. http://groups.google.com/group/comp.text.tex/msg/fbf7d61a0c3a807d

[4] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:idbo94uka1@four.albasani.net, 2010-12-03. http://groups.google.com/group/comp.text.tex/msg/0c230ee479487962

[5] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:idbpu4$cg1$1@news.albasani.net, 2010-12-03.
http://groups.google.com/group/comp.text.tex/msg/bbef4263390d647b

[6] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:idd4ga$r83$1@four.albasani.net, 2010-12-04.
http://groups.google.com/group/comp.text.tex/msg/00dfd1ec103cd272

[7] GL: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:4cfa2e27$0$7389$426a74cc@news.free.fr, 2010-12-04.
http://groups.google.com/group/comp.text.tex/msg/d3a75995c1cf267e

[8] Heiko Oberdiek: *Re: TeX refuses to strip outer braces in argument*; newsgroup comp.text.tex, news:iddhq1$3kj$1@news.eternal-september.org, 2010-12-04.
http://groups.google.com/group/comp.text.tex/msg/5f7a23e3ab70e347

[9] David Kastrup: *How to detect if* `\vbox` *is empty*; newsgroup comp.text.tex, 2011-02-04.
http://groups.google.com/group/comp.text.tex/msg/8d3cb89496a4d86d

6 History

[2009/08/05 v1.0]

- First version.

[2009/12/12 v1.1]

- Short title shortened.
- `\ltx@IfUndefined` added.

[2010/01/28 v1.2]

- `\ltx@RemovePrefix` and `\ltx@StripPrefix` added.
- `\ltx@ifclassloaded`, `\ltx@ifpackageloaded`, `\ltx@iffileloaded`, `\ltx@ifclasslater`, `\ltx@ifpackagelater`, `\ltx@iffilelater`, `\ltx@clsextension`, `\ltx@pkgextension` added.
- `\ltx@GlobalAppendToMacro`, `\ltx@LocalAppendToMacro` added.

[2010/03/01 v1.3]

- `\ltx@newif` added.
- `\ltx@ifnextchar` added.
- Numbers `\ltx@zero`, `\ltx@one`, `\ltx@two`, `\ltx@cclv` added.

[2010/03/09 v1.4]

- `\ltx@pkgextension` and `\ltx@clsextension` are hardcoded to avoid trouble with `\@onlypreamble`.

[2010/04/08 v1.5]

- `\ltx@cartwo`, `\ltx@cdrtwo`, `\ltx@carthree`, `\ltx@cdrthree`, `\ltx@carfour`, `\ltx@cdrfour` added.
- `\ltx@ReturnAfterFi` and `\ltx@ReturnAfterElseFi` fixed.

[2010/04/16 v1.6]

- `\ltx@leavevmode`, `\ltx@mbox` added.

[2010/04/26 v1.7]

- `\ltx@GobbleNum`, `\ltx@CdrNum`, `\ltx@CarNum` added.
- `\ltx@carzero`, `\ltx@cdrzero` added.
- `\ltx@hashchar` added.

[2010/09/11 v1.8]

- `\ltx@leftbracechar`, `\ltx@rightbracechar` added.

[2010/10/25 v1.9]

- `\ltx@LocalAppendToMacro` and `\ltx@GlobalAppendToMacro` are now `\long`.

[2010/10/31 v1.10]

- `\ltx@newglobalif` added.

[2010/11/12 v1.11]

- `\ltx@ifempty` added.
- `\ltx@firstofthree`, `\ltx@secondofthree`, `\ltx@thirdofthree` added.

[2010/12/02 v1.12]

- `\ltx@onelevel@sanitize` added.
- `\LTXcmds@num` fixed for the case with `\numexpr` (bug found by GL).

[2010/12/04 v1.13]

- `\ltx@ifblank` added.
- Optimization for `\ltx@ifempty`.

[2010/12/07 v1.14]

- `\ltx@zapspace` added.

[2010/12/12 v1.15]

- `\ltx@minusone` added.

[2011/02/04 v1.16]

- `\ltx@IfBoxEmpty` and `\ltx@IfBoxVoidOrEmpty` added.
- `\ltx@firstoffour`, ..., `\ltx@fourthoffour` added.

[2011/02/05 v1.17]

- `\ltx@IfBoxEmpty`: an empty box may have non-zero dimensions.

[2011/03/16 v1.18]

- `\ltx@ifclasslater` fixed.

7 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

L