

The ltxcmds package

Heiko Oberdiek
<heiko.oberdiek at googlemail.com>

2011/04/18 v1.20

Abstract

The package `ltxcmds` exports some utility macros from the L^AT_EX kernel into a separate namespace and also provides them for other formats such as plain-T_EX.

Contents

1	Documentation	3
1.1	Introduction	3
1.2	Numbers	3
1.3	Scratch registers	3
1.4	Argument killers	3
1.5	Argument grabbers	4
1.6	List helpers	4
1.7	Tail recursion	5
1.8	Empty macro	5
1.9	Characters	5
1.10	Boolean switch	5
1.11	Command definitions	5
1.12	Stripping	6
1.13	File management	6
1.13.1	File extensions	6
1.13.2	Load check	6
1.13.3	Version date check	7
1.14	Macro additions	7
1.15	Next character detection	7
1.16	<code>\ltx@leavevmode</code> , <code>\ltx@mbox</code>	8
1.17	Expandable test for emptiness	8
1.18	Stripping spaces	8
1.19	Check for emptiness of boxes	8
2	Implementation	9
2.1	Identification	9
2.2	Numbers	11
2.3	Scratch registers	11
2.4	Argument killers	13
2.5	Argument grabbers	13
2.6	List helpers	14
2.7	Tail recursion	15
2.8	Empty macro	15
2.9	Characters	15
2.10	Boolean switch	16
2.11	Command definitions	17
2.12	Stripping	18

2.13	File management	18
2.13.1	File extensions	18
2.13.2	Load check	18
2.13.3	Version date check	19
2.14	Macro additions	20
2.15	Next character detection	20
2.16	<code>\ltx@leavevmode</code> , <code>\ltx@mbox</code>	21
2.17	Help macros	22
2.18	Expandable test for emptiness	22
2.18.1	Vanilla <code>T_EX</code>	22
2.18.2	With <code>\detokenize</code>	23
2.18.3	<code>\ltx@ifblank</code>	23
2.19	<code>\ltx@zapspace</code>	24
2.20	<code>\ltx@ifBoxEmpty</code>	24
3	Test	25
3.1	Catcode checks for loading	25
3.2	Test <code>\ltx@GobbleNum</code>	27
3.3	Test <code>\ltx@ifempty</code>	30
3.4	Test <code>\ltx@zap@space</code>	32
3.5	Test <code>\ltx@ifBoxEmpty</code>	33
3.6	Test for next character detection	34
4	Installation	37
4.1	Download	37
4.2	Bundle installation	37
4.3	Package installation	38
4.4	Refresh file name databases	38
4.5	Some details for the interested	38
5	References	39
6	History	39
	[2009/08/05 v1.0]	39
	[2009/12/12 v1.1]	39
	[2010/01/28 v1.2]	39
	[2010/03/01 v1.3]	40
	[2010/03/09 v1.4]	40
	[2010/04/08 v1.5]	40
	[2010/04/16 v1.6]	40
	[2010/04/26 v1.7]	40
	[2010/09/11 v1.8]	40
	[2010/10/25 v1.9]	40
	[2010/10/31 v1.10]	40
	[2010/11/12 v1.11]	40
	[2010/12/02 v1.12]	40
	[2010/12/04 v1.13]	40
	[2010/12/07 v1.14]	41
	[2010/12/12 v1.15]	41
	[2011/02/04 v1.16]	41
	[2011/02/05 v1.17]	41
	[2011/03/16 v1.18]	41
	[2011/04/14 v1.19]	41
	[2011/04/18 v1.20]	41
7	Index	41

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	\rightarrow	0
\ltx@one	\rightarrow	1
\ltx@two	\rightarrow	2
\ltx@ccclv	\rightarrow	255
\ltx@minusone	\rightarrow	-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 Scratch registers

Following the conventions of plain \TeX and \LaTeX the first ten registers are free to use. Even numbered registers are for local, odd numbered for global use.

$\text{\ltx@}(\text{Loc},\text{Glob})(\text{Toks},\text{Dimen},\text{Skip})(\text{A},\text{B},\text{C},\text{D},\text{E})$
--

The name consists of the prefix \ltx@ , then **Loc** or **Glob** for local or global usage follows. The register type is given by **Toks** for token register, **Dimen** for dimen register and **Skip** for skip register. As last part the registers are numbered from **A** to **E**. Example: \ltx@LocToksA .

Since 2011/04/14 v1.19.

1.4 Argument killers

$\text{\ltx@gobble} \{\langle 1 \rangle\}$	\rightarrow
$\text{\ltx@gobbletwo} \{\langle 1 \rangle\} \{\langle 2 \rangle\}$	\rightarrow
$\text{\ltx@gobblethree} \{\langle 1 \rangle\} \{\langle 2 \rangle\} \{\langle 3 \rangle\}$	\rightarrow
$\text{\ltx@gobblefour} \{\langle 1 \rangle\} \{\langle 2 \rangle\} \{\langle 3 \rangle\} \{\langle 4 \rangle\}$	\rightarrow

$\text{\ltx@GobbleNum} \{\langle num \rangle\} \{\langle 1 \rangle\} \{\langle 2 \rangle\} \dots \{\langle \langle num \rangle \rangle\}$	\rightarrow
---	---------------

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

1.5 Argument grabbers

<code>\ltx@firstofone {⟨1⟩}</code>	\rightarrow	$\langle 1 \rangle$
<code>\ltx@firstoftwo {⟨1⟩} {⟨2⟩}</code>	\rightarrow	$\langle 1 \rangle$
<code>\ltx@secondoftwo {⟨1⟩} {⟨2⟩}</code>	\rightarrow	$\langle 2 \rangle$
<code>\ltx@firstofthree {⟨1⟩} {⟨2⟩} {⟨3⟩}</code>	\rightarrow	$\langle 1 \rangle$
<code>\ltx@secondofthree {⟨1⟩} {⟨2⟩} {⟨3⟩}</code>	\rightarrow	$\langle 2 \rangle$
<code>\ltx@thirdofthree {⟨1⟩} {⟨2⟩} {⟨3⟩}</code>	\rightarrow	$\langle 3 \rangle$
<code>\ltx@firstoffour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩}</code>	\rightarrow	$\langle 1 \rangle$
<code>\ltx@secondoffour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩}</code>	\rightarrow	$\langle 2 \rangle$
<code>\ltx@thirdoffour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩}</code>	\rightarrow	$\langle 3 \rangle$
<code>\ltx@fourthoffour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩}</code>	\rightarrow	$\langle 4 \rangle$

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

1.6 List helpers

<code>\ltx@carzero ... \@nil</code>	\rightarrow	
<code>\ltx@cdrzero ... \@nil</code>	\rightarrow	...

<code>\ltx@car {⟨1⟩} ... \@nil</code>	\rightarrow	$\langle 1 \rangle$
<code>\ltx@cdr {⟨1⟩} ... \@nil</code>	\rightarrow	...

<code>\ltx@cartwo {⟨1⟩} {⟨2⟩} ... \@nil</code>	\rightarrow	$\langle 1 \rangle \langle 2 \rangle$
<code>\ltx@cdrtwo {⟨1⟩} {⟨2⟩} ... \@nil</code>	\rightarrow	...

<code>\ltx@carthree {⟨1⟩} {⟨2⟩} {⟨3⟩} ... \@nil</code>	\rightarrow	$\langle 1 \rangle \langle 2 \rangle \langle 3 \rangle$
<code>\ltx@cdrthree {⟨1⟩} {⟨2⟩} {⟨3⟩} ... \@nil</code>	\rightarrow	...

<code>\ltx@carfour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩} ... \@nil</code>	\rightarrow	$\langle 1 \rangle \langle 2 \rangle \langle 3 \rangle \langle 4 \rangle$
<code>\ltx@cdrfour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩} ... \@nil</code>	\rightarrow	...

<code>\ltx@CarNum {⟨num⟩} {⟨1⟩} ... {⟨⟨num⟩⟩} {⟨⟨num⟩+1⟩} ... \@nil</code>	\rightarrow	$\{ \langle 1 \rangle \} \dots \{ \langle \langle num \rangle \rangle \} \dots$
<code>\ltx@CdrNum {⟨num⟩} {⟨1⟩} ... {⟨⟨num⟩⟩} {⟨⟨num⟩+1⟩} ... \@nil</code>	\rightarrow	$\{ \langle \langle num \rangle + 1 \rangle \} \dots$

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

1.7 Tail recursion

<code>\ltx@ReturnAfterFi {⟨1⟩} \fi</code>	\rightarrow	<code>\fi ⟨1⟩</code>
<code>\ltx@ReturnAfterElseFi {⟨1⟩} \else {⟨2⟩} \fi</code>	\rightarrow	<code>\fi ⟨1⟩</code>

1.8 Empty macro

<code>\ltx@empty</code>	\rightarrow
-------------------------	---------------

1.9 Characters

<code>\ltx@space</code>	\rightarrow	<code>\space</code>
<code>\ltx@percentchar</code>	\rightarrow	<code>\%</code>
<code>\ltx@backslashchar</code>	\rightarrow	<code>\</code>
<code>\ltx@hashchar</code>	\rightarrow	<code>\#</code> (since v1.7)
<code>\ltx@leftbracechar</code>	\rightarrow	<code>\{</code> (since v1.8)
<code>\ltx@rightbracechar</code>	\rightarrow	<code>\}</code> (since v1.8)

1.10 Boolean switch

<code>\ltx@newif {⟨cmd⟩}</code>

`\ltx@newif` defines a new boolean switch `⟨cmd⟩` like `\newif`. Unlike plain $\mathrm{T}_{\mathrm{E}}\mathrm{X}$'s `\newif`, `\ltx@newif` is not `\outer`. The command `⟨cmd⟩` must start with the two characters `if`.

<code>\ltx@newglobalif {⟨cmd⟩}</code>

`\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.

1.11 Command definitions

<code>\ltx@ifundefined {⟨cmd⟩} {⟨yes⟩} {⟨no⟩}</code>
--

If $\varepsilon\text{-T}_{\mathrm{E}}\mathrm{X}$ 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”.

<code>\ltx@ifUndefined {⟨cmd⟩} {⟨yes⟩} {⟨no⟩}</code>
--

If $\varepsilon\text{-T}_{\mathrm{E}}\mathrm{X}$ 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 $\varepsilon\text{-T}_{\mathrm{E}}\mathrm{X}$.

`\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.

1.12 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 L^AT_EX’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 L^AT_EX’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.

1.13 File management

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

1.13.1 File extensions

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

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

1.13.2 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 L^AT_EX’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 L^AT_EX 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 be specified in the latter command and vice versa.

1.13.3 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.

1.14 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.

1.15 Next character detection

```
\ltx@ifnextchar {<char>} {<yes>} {<no>}
```

If next character is `<char>` then `<yes>` is called, otherwise `<no>`. The character is not removed. Spaces are silently removed when looking for `<char>` as \LaTeX 's version `\kernel@ifnextchar` does. But there are also small differences:

- The space can be used as `<char>`. In this case optional spaces before `<char>` are not supported of course.
- If the optional space is a command that is a character (defined by `\let` or `\futurelet`), then `\kernel@ifnextchar` breaks with an \TeX error. `\ltx@ifnextchar` silently removes this token as optional space.

Since 2010/03/01 v1.3.

```
\ltx@ifnextchar@nospace {<char>} {<yes>} {<no>}
```

Macro `\ltx@ifnextchar@nospace` behaves like macro `\ltx@ifnextchar` with the exception that optional spaces are not supported before `<char>`. Since 2011/04/14 v1.19.

1.16 `\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`.

1.17 Expandable test for emptiness

`\ltx@ifempty` $\{\langle stuff \rangle\}$ $\{\langle yes \rangle\}$ $\{\langle no \rangle\}$

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

`\ltx@ifblank` $\{\langle stuff \rangle\}$ $\{\langle yes \rangle\}$ $\{\langle no \rangle\}$

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

1.18 Stripping spaces

`\ltx@zapspace` $\{\langle stuff \rangle\}$

Macro `\ltx@zapspace` strips spaces from $\langle stuff \rangle$ that are not hidden inside curly braces. Like L^AT_EX's `\zap@space` it is expandable. Differences:

- Syntax: `\zap@space` also expects a space token and `\@empty` after $\langle stuff \rangle$.
- 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.

1.19 Check for emptiness of boxes

`\ltx@ifBoxEmpty` $\{\langle box\ register\ number \rangle\}$ $\{\langle yes \rangle\}$ $\{\langle no \rangle\}$

Macro `\ltx@ifBoxEmpty` calls $\langle yes \rangle$ 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 $\langle no \rangle$ 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 0.0pt, but the box does not contain anything. Macro `\ltx@ifboxempty` is available since 2010/02/04 v1.16.

`\ltx@ifboxvoidorelse {<box register number>} {<yes>} {<no>}`

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

2 Implementation

2.1 Identification

```
1 <*package>
```

Reload check, especially if the package is not used with L^AT_EX.

```
2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^^M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@ltxcmds.sty\endcsname
15 \ifx\x\relax % plain-TeX, first loading
16 \else
17 \def\empty{}%
18 \ifx\x\empty % LaTeX, first loading,
19 % variable is initialized, but \ProvidesPackage not yet seen
20 \else
21 \expandafter\ifx\csname PackageInfo\endcsname\relax
22 \def\x#1#2{%
23 \immediate\write-1{Package #1 Info: #2.}%
24 }%
25 \else
26 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
27 \fi
28 \x{ltxcmds}{The package is already loaded}%
29 \aftergroup\endinput
30 \fi
31 \fi
32 \endgroup%
```

Package identification:

```
33 \begingroup\catcode61\catcode48\catcode32=10\relax%
34 \catcode13=5 % ^^M
35 \endlinechar=13 %
36 \catcode35=6 % #
37 \catcode39=12 % '
38 \catcode40=12 % (
39 \catcode41=12 % )
40 \catcode44=12 % ,
41 \catcode45=12 % -
42 \catcode46=12 % .
43 \catcode47=12 % /
44 \catcode58=12 % :
```

```

45 \catcode64=11 % @
46 \catcode91=12 % [
47 \catcode93=12 % ]
48 \catcode123=1 % {
49 \catcode125=2 % }
50 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
51   \def\x#1#2#3[#4]{\endgroup
52     \immediate\write-1{Package: #3 #4}%
53     \xdef#1{#4}%
54   }%
55 \else
56   \def\x#1#2[#3]{\endgroup
57     #2[{#3}]%
58     \ifx#1\@undefined
59       \xdef#1{#3}%
60     \fi
61     \ifx#1\relax
62       \xdef#1{#3}%
63     \fi
64   }%
65 \fi
66 \expandafter\x\csname ver@ltxcmds.sty\endcsname
67 \ProvidesPackage{ltxcmds}%
68 [2011/04/18 v1.20 LaTeX kernel commands for general use (H0)]%
69 \begingroup\catcode61\catcode48\catcode32=10\relax%
70 \catcode13=5 % ^^M
71 \endlinechar=13 %
72 \catcode123=1 % {
73 \catcode125=2 % }
74 \catcode64=11 % @
75 \def\x{\endgroup
76   \expandafter\edef\csname LTXcmds@AtEnd\endcsname{%
77     \endlinechar=\the\endlinechar\relax
78     \catcode13=\the\catcode13\relax
79     \catcode32=\the\catcode32\relax
80     \catcode35=\the\catcode35\relax
81     \catcode61=\the\catcode61\relax
82     \catcode64=\the\catcode64\relax
83     \catcode123=\the\catcode123\relax
84     \catcode125=\the\catcode125\relax
85   }%
86 }%
87 \x\catcode61\catcode48\catcode32=10\relax%
88 \catcode13=5 % ^^M
89 \endlinechar=13 %
90 \catcode35=6 % #
91 \catcode64=11 % @
92 \catcode123=1 % {
93 \catcode125=2 % }
94 \def\TMP@EnsureCode#1#2{%
95   \edef\LTXcmds@AtEnd{%
96     \LTXcmds@AtEnd
97     \catcode#1=\the\catcode#1\relax
98   }%
99   \catcode#1=#2\relax
100 }
101 \TMP@EnsureCode{36}{3}% $
102 \TMP@EnsureCode{38}{4}% &
103 \TMP@EnsureCode{40}{12}% (
104 \TMP@EnsureCode{41}{12}% )
105 \TMP@EnsureCode{45}{12}% -
106 \TMP@EnsureCode{46}{12}% .

```

```

107 \TMP@EnsureCode{47}{12}% /
108 \TMP@EnsureCode{60}{12}% <
109 \TMP@EnsureCode{62}{12}% >
110 \TMP@EnsureCode{91}{12}% [
111 \TMP@EnsureCode{96}{12}% ‘
112 \TMP@EnsureCode{93}{12}% ]
113 \TMP@EnsureCode{94}{12}% ^ (superscript) (!)
114 \TMP@EnsureCode{124}{12}% |
115 \edef\LTXcmds@AtEnd{\LTXcmds@AtEnd\noexpand\endinput}

```

2.2 Numbers

```

\ltx@zero

116 \chardef\ltx@zero=0 %

\ltx@one

117 \chardef\ltx@one=1 %

\ltx@two

118 \chardef\ltx@two=2 %

\ltx@active

119 \chardef\ltx@active=13 %

\ltx@ccclv

120 \chardef\ltx@ccclv=255 %

\ltx@minusone

121 \def\ltx@minusone{%
122   -\ltx@one
123 }

```

2.3 Scratch registers

```

\ltx@LocToksA

124 \toksdef\ltx@LocToksA=0 %

\ltx@LocToksB

125 \toksdef\ltx@LocToksB=2 %

\ltx@LocToksC

126 \toksdef\ltx@LocToksC=4 %

\ltx@LocToksD

127 \toksdef\ltx@LocToksD=6 %

\ltx@LocToksE

128 \toksdef\ltx@LocToksE=8 %

\ltx@GlobToksA

129 \toksdef\ltx@GlobToksA=1 %

\ltx@GlobToksB

130 \toksdef\ltx@GlobToksB=3 %

\ltx@GlobToksC

131 \toksdef\ltx@GlobToksC=5 %

```

```

\ltx@GlobToksD
132 \toksdef\ltx@GlobToksD=7 %

\ltx@GlobToksE
133 \toksdef\ltx@GlobToksE=9 %

\ltx@LocDimenA
134 \dimendef\ltx@LocDimenA=0 %

\ltx@LocDimenB
135 \dimendef\ltx@LocDimenB=2 %

\ltx@LocDimenC
136 \dimendef\ltx@LocDimenC=4 %

\ltx@LocDimenD
137 \dimendef\ltx@LocDimenD=6 %

\ltx@LocDimenE
138 \dimendef\ltx@LocDimenE=8 %

\ltx@GlobDimenA
139 \dimendef\ltx@GlobDimenA=1 %

\ltx@GlobDimenB
140 \dimendef\ltx@GlobDimenB=3 %

\ltx@GlobDimenC
141 \dimendef\ltx@GlobDimenC=5 %

\ltx@GlobDimenD
142 \dimendef\ltx@GlobDimenD=7 %

\ltx@GlobDimenE
143 \dimendef\ltx@GlobDimenE=9 %

\ltx@LocSkipA
144 \skipdef\ltx@LocSkipA=0 %

\ltx@LocSkipB
145 \skipdef\ltx@LocSkipB=2 %

\ltx@LocSkipC
146 \skipdef\ltx@LocSkipC=4 %

\ltx@LocSkipD
147 \skipdef\ltx@LocSkipD=6 %

\ltx@LocSkipE
148 \skipdef\ltx@LocSkipE=8 %

\ltx@GlobSkipA
149 \skipdef\ltx@GlobSkipA=1 %

\ltx@GlobSkipB
150 \skipdef\ltx@GlobSkipB=3 %

```

```

\ltx@GlobSkipC
151 \skipdef\ltx@GlobSkipC=5 %

\ltx@GlobSkipD
152 \skipdef\ltx@GlobSkipD=7 %

\ltx@GlobSkipE
153 \skipdef\ltx@GlobSkipE=9 %

```

2.4 Argument killers

```

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

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

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

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

\ltx@GobbleNum
158 \def\ltx@GobbleNum#1{%
159   \romannumeral
160   \csname ltx@zero%
161   \expandafter\LTxcmds@GobbleNum
162   \romannumeral\LTxcmds@num{#1}000{m\endcsname}%
163 }

\LTxcmds@GobbleNum
164 \def\LTxcmds@GobbleNum#1{%
165   \csname LTxcmds@G#1\LTxcmds@GobbleNum
166 }

\LTxcmds@Gm
167 \long\def\LTxcmds@Gm#1{%
168   \endcsname
169 }

```

2.5 Argument grabbers

```

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

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

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

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

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

```

```

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

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

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

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

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

```

2.6 List helpers

```

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

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

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

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

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

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

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

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

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

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

\ltx@CarNum
190 \def\ltx@CarNum#1{%
191   \romannumeral
192   \csname LTXcmds@CarNumFinish%
193   \expandafter\LTXcmds@CarNum
194   \romannumeral\LTXcmds@num{#1}000{x\endcsname}%
195 }

\LTXcmds@CarNum
196 \def\LTXcmds@CarNum#1{%
197   \csname LTXcmds@C#1\LTXcmds@CarNum
198 }

```

\LTXcmds@Cm

```
199 \long\def\LTXcmds@Cm#1#2{%
200   \endcsname{#1#2}%
201 }
```

\LTXcmds@Cx

```
202 \def\LTXcmds@Cx#1{%
203   \endcsname{}%
204 }
```

\LTXcmds@CarNumFinish

```
205 \long\def\LTXcmds@CarNumFinish#1#2\@nil{%
206   \ltx@zero
207   #1%
208 }
```

\ltx@CdrNum

```
209 \def\ltx@CdrNum#1{%
210   \romannumeral0%
211   \expandafter\expandafter\expandafter\LTXcmds@CdrNum
212   \ltx@GobbleNum{#1}%
213 }
```

\LTXcmds@CdrNum

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

2.7 Tail recursion

\ltx@ReturnAfterFi

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

\ltx@ReturnAfterElseFi

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

2.8 Empty macro

\ltx@empty

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

2.9 Characters

\ltx@space

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

\ltx@percentchar

```
219 \begingroup
220   \lccode'0='%\relax
221 \lowercase{\endgroup
222   \def\ltx@percentchar{0}%
223 }
```

\ltx@backslashchar

```
224 \begingroup
225   \lccode'0='\\ \relax
226 \lowercase{\endgroup
227   \def\ltx@backslashchar{0}%
228 }
```

`\ltx@hashchar`

```
229 \begingroup
230   \lccode'0='#\relax
231 \lowercase{\endgroup
232   \def\ltx@hashchar{0}%
233 }
```

`\ltx@leftbracechar`

```
234 \begingroup
235   \lccode'0='{ \relax
236 \lowercase{\endgroup
237   \def\ltx@leftbracechar{0}%
238 }
```

`\ltx@rightbracechar`

```
239 \begingroup
240   \lccode'0='}\relax
241 \lowercase{\endgroup
242   \def\ltx@rightbracechar{0}%
243 }
```

2.10 Boolean switch

`\ltx@newif`

```
244 \def\ltx@newif#1{%
245   \begingroup
246     \escapechar=-1 %
247   \expandafter\endgroup
248   \expandafter\LTXcmds@newif\string#1\@nil
249 }
```

`\LTXcmds@newif`

```
250 \begingroup
251   \escapechar=-1 %
252 \expandafter\endgroup
253 \expandafter\def\expandafter\LTXcmds@newif\string\if#1\@nil{%
254   \expandafter\edef\csname#1true\endcsname{%
255     \let
256     \expandafter\noexpand\csname if#1\endcsname
257     \noexpand\iftrue
258   }%
259   \expandafter\edef\csname#1false\endcsname{%
260     \let
261     \expandafter\noexpand\csname if#1\endcsname
262     \noexpand\iffalse
263   }%
264   \csname#1false\endcsname
265 }
```

`\ltx@newglobalif`

```
266 \def\ltx@newglobalif#1{%
267   \begingroup
268     \escapechar=-1 %
269   \expandafter\endgroup
270   \expandafter\LTXcmds@newglobalif\string#1\@nil
271 }
```

`\LTXcmds@newglobalif`

```
272 \begingroup
273   \escapechar=-1 %
```



```

274 \expandafter\endgroup
275 \expandafter
276 \def\expandafter\LTXcmds@newglobalif\string\if#1\@nil{%
277   \expandafter\edef\csname#1true\endcsname{%
278     \global\let
279     \expandafter\noexpand\csname if#1\endcsname
280     \noexpand\iftrue
281   }%
282   \expandafter\edef\csname#1false\endcsname{%
283     \global\let
284     \expandafter\noexpand\csname if#1\endcsname
285     \noexpand\iffalse
286   }%
287   \csname#1false\endcsname
288 }

```

2.11 Command definitions

\ltx@LocalExpandAfter

```

289 \def\ltx@LocalExpandAfter{%
290   \begingroup
291     \expandafter\expandafter\expandafter
292   \endgroup
293   \expandafter
294 }

295 \ltx@LocalExpandAfter
296 \ifx\csname ifcsname\endcsname\relax

```

\ltx@ifundefined

```

297 \def\ltx@ifundefined#1{%
298   \expandafter\ifx\csname #1\endcsname\relax
299     \expandafter\ltx@firstoftwo
300   \else
301     \expandafter\ltx@secondoftwo
302   \fi
303 }%

```

\ltx@ifUndefined

```

304 \def\ltx@ifUndefined#1{%
305   \begingroup\expandafter\expandafter\expandafter\endgroup
306   \expandafter\ifx\csname #1\endcsname\relax
307     \expandafter\ltx@firstoftwo
308   \else
309     \expandafter\ltx@secondoftwo
310   \fi
311 }%

312 \expandafter\ltx@gobble
313 \else
314 \expandafter\ltx@firstofone
315 \fi
316 {%

```

\ltx@ifundefined

```

317 \def\ltx@ifundefined#1{%
318   \ifcsname #1\endcsname
319     \expandafter\ifx\csname #1\endcsname\relax
320       \expandafter\expandafter\expandafter\ltx@firstoftwo
321     \else
322       \expandafter\expandafter\expandafter\ltx@secondoftwo
323     \fi

```

```

324     \else
325         \expandafter\ltx@firstoftwo
326     \fi
327 }%

```

\ltx@ifundefined

```

328 \let\ltx@ifundefined\ltx@ifundefined
329 }

```

2.12 Stripping

\ltx@RemovePrefix

```

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

```

\ltx@StripPrefix

```

331 \def\ltx@StripPrefix{%
332     \expandafter\ltx@RemovePrefix
333 }

```

\ltx@onelevel@sanitize

```

334 \def\ltx@onelevel@sanitize#1{%
335     \edef#1{%
336         \expandafter
337         \ltx@RemovePrefix\meaning#1%
338     }%
339 }

```

2.13 File management

2.13.1 File extensions

\ltx@clsextension

```

340 \def\ltx@clsextension{cls}

```

\ltx@pkgextension

```

341 \def\ltx@pkgextension{sty}

```

2.13.2 Load check

\ltx@iffileloaded

```

342 \def\ltx@iffileloaded#1{%
343     \ltx@ifundefined{ver@#1}\ltx@secondoftwo\ltx@firstoftwo
344 }

```

\ltx@ifclassloaded

```

345 \def\ltx@ifclassloaded#1{%
346     \ltx@iffileloaded{#1.\ltx@clsextension}%
347 }

```

\ltx@ifpackageloaded

```

348 \def\ltx@ifpackageloaded#1{%
349     \ltx@iffileloaded{#1.\ltx@pkgextension}%
350 }

```

2.13.3 Version date check

\ltx@iffilelater

```
351 \def\ltx@iffilelater#1#2{%
352   \ltx@iffileloaded{#1}{%
353     \expandafter\LTXcmds@IfLater\expandafter{%
354       \number
355       \expandafter\expandafter\expandafter\LTXcmds@ParseVersion
356       \expandafter\expandafter\expandafter{%
357         \csname ver@#1\endcsname
358       }%
359     \expandafter}\expandafter{%
360       \number
361       \expandafter\LTXcmds@ParseVersion\expandafter{#2}%
362     }%
363   }\ltx@secondoftwo
364 }
```

\LTXcmds@IfLater

```
365 \def\LTXcmds@IfLater#1#2{%
366   \ifcase 0%
367     \ifnum#1<19940101 %
368   \else
369     \ifnum#2<19940101 %
370   \else
371     \ifnum#2>#1 %
372   \else
373     1%
374   \fi
375   \fi
376   \fi
377   \ltx@space
378   \expandafter\ltx@secondoftwo
379 \else
380   \expandafter\ltx@firstoftwo
381 \fi
382 }
```

\ltx@ifclasslater

```
383 \def\ltx@ifclasslater#1{%
384   \ltx@iffilelater{#1.\ltx@clsextension}%
385 }
```

\ltx@ifpackagelater

```
386 \def\ltx@ifpackagelater#1{%
387   \ltx@iffilelater{#1.\ltx@pkgextension}%
388 }
```

```
389 \ltx@ifUndefined{pdfmatch}{%}
```

\LTXcmds@ParseVersion

```
390 \def\LTXcmds@ParseVersion#1{%
391   \LTXcmds@@ParseVersion#10000/00/00\@nil
392   }%
```

\LTXcmds@@ParseVersion

```
393 \def\LTXcmds@@ParseVersion#1#2#3#4/#5#6/#7#8#9\@nil{%
394   #1#2#3#4#5#6#7#8%
395   }%
396 }
```

\LTXcmds@ParseVersion

```
397 \def\LTXcmds@ParseVersion#1{%
398   \ifnum\pdfmatch{%
399     ~%
400     (199[4-9] | [2-9] [0-9] [0-9] [0-9] )/%
401     (0[1-9] | 1[0-2] )/%
402     (0[1-9] | [1-2] [0-9] | 3[0-1] )%
403   }{#1}=1 %
404   \ltx@StripPrefix\pdflastmatch1 %
405   \ltx@StripPrefix\pdflastmatch2 %
406   \ltx@StripPrefix\pdflastmatch3 %
407   \else
408     0%
409   \fi
410 }%
411 }
```

2.14 Macro additions

\ltx@GlobalAppendToMacro

```
412 \long\def\ltx@GlobalAppendToMacro#1#2{%
413   \ifx\ltx@undefined#1%
414     \let#1\ltx@empty
415   \else
416     \ifx\relax#1%
417       \let#1\ltx@empty
418     \fi
419   \fi
420   \begingroup
421     \ltx@LocToksA\expandafter{#1#2}%
422     \xdef#1{\the\ltx@LocToksA}%
423   \endgroup
424 }
```

\ltx@LocalAppendToMacro

```
425 \long\def\ltx@LocalAppendToMacro#1#2{%
426   \global\let\LTXcmds@gtemp#1%
427   \ifx\ltx@undefined\LTXcmds@gtemp
428     \global\let\LTXcmds@gtemp\ltx@empty
429   \else
430     \ifx\relax\LTXcmds@gtemp
431       \global\let\LTXcmds@gtemp\ltx@empty
432     \fi
433   \fi
434   \begingroup
435     \ltx@LocToksA\expandafter{\LTXcmds@gtemp#2}%
436     \xdef\LTXcmds@gtemp{\the\ltx@LocToksA}%
437   \endgroup
438   \let#1\LTXcmds@gtemp
439 }
```

2.15 Next character detection

\ltx@ifnextchar

```
440 \long\def\ltx@ifnextchar#1#2#3{%
441   \begingroup
442   \let\LTXcmds@CharToken= #1\relax
443   \ltx@LocToksA{\endgroup#2}%
444   \ltx@LocToksB{\endgroup#3}%
445   \futurelet\LTXcmds@LetToken\LTXcmds@ifnextchar
446 }
```

\LTXcmds@ifnextchar

```

447 \def\LTXcmds@ifnextchar{%
448   \ifx\LTXcmds@LetToken\LTXcmds@CharToken
449     \the\expandafter\ltx@LocToksA
450   \else
451     \expandafter
452       \ifx\csname LTXcmds@LetToken\endcsname\LTXcmds@SpaceToken
453       \expandafter\expandafter\expandafter\LTXcmds@ifnextchar
454     \else
455       \the\expandafter\expandafter\expandafter\ltx@LocToksB
456     \fi
457   \fi
458 }

```

\LTXcmds@ifnextchar \futurelet does not distinguish between a character and a command that is a character (defined by using \let or \futurelet). Therefore the space is caught by \romannumeral with negative character constant that gobbles one optional space.

```

459 \def\LTXcmds@ifnextchar{%
460   \expandafter\futurelet
461   \expandafter\LTXcmds@LetToken
462   \expandafter\LTXcmds@ifnextchar
463   \romannumeral-'\.%
464 }

```

\LTXcmds@SpaceToken

```

465 \ltx@firstofone{\let\LTXcmds@SpaceToken= } %

```

\ltx@ifnextchar@nospace

```

466 \long\def\ltx@ifnextchar@nospace#1#2#3{%
467   \begingroup
468   \let\LTXcmds@CharToken= #1\relax
469   \ltx@LocToksA{\endgroup#2}%
470   \ltx@LocToksB{\endgroup#3}%
471   \futurelet\LTXcmds@LetToken\LTXcmds@ifnextchar@nospace
472 }

```

\LTXcmds@ifnextchar@nospace

```

473 \def\LTXcmds@ifnextchar@nospace{%
474   \the
475   \ifx\LTXcmds@LetToken\LTXcmds@CharToken
476     \expandafter\ltx@LocToksA
477   \else
478     \expandafter\ltx@LocToksB
479   \fi
480 }

```

2.16 \ltx@leavevmode, \ltx@mbox

\ltx@leavevmode

```

481 \ltx@ifundefined{quitvmode}{%
482   \ltx@ifundefined{leavevmode}{%
483     \ltx@ifundefined{voidb@x}{%
484       \ltx@ifundefined{newbox}{%
485         \def\ltx@leavevmode{%
486           \begingroup
487             \setbox\ltx@zero=\hbox{}}%
488           \begingroup
489             \setbox\ltx@zero=\hbox{\box\ltx@zero}%
490           \endgroup
491           \unhbox\ltx@zero

```

```

492         \endgroup
493     }%
494 }{%
495     \csname newbox\endcsname\LTxcmds@VoidBox
496     \ifvoid\LTxcmds@VoidBox
497     \else
498         \setbox\LTxcmds@VoidBox=\hbox{}%
499         \begingroup
500             \setbox\LTxcmds@VoidBox=\hbox{\box\LTxcmds@VoidBox}%
501         \endgroup
502     \fi
503     \def\ltx@leavevmode{\unhbox\LTxcmds@VoidBox}%
504 }%
505 }{%
506     \def\ltx@leavevmode{\unhbox\voidb@x}%
507 }%
508 }{%
509     \let\ltx@leavevmode\leavevmode
510 }%
511 }{%
512     \let\ltx@leavevmode\quitvmode
513 }

```

`\ltx@mbox`

```

514 \def\ltx@mbox{%
515     \ltx@leavevmode
516     \hbox
517 }

```

2.17 Help macros

`\LTxcmds@num`

```

518 \ltx@ifundefined{numexpr}{%
519     \def\LTxcmds@num#1{%
520         \expandafter\ltx@firstofone\expandafter{%
521             \number#1%
522         }%
523     }%
524 }{%
525     \def\LTxcmds@num#1{%
526         \expandafter\ltx@firstofone\expandafter{%
527             \the\numexpr#1%
528         }%
529     }%
530 }

```

2.18 Expandable test for emptiness

```

531 \ltx@ifundefined{detokenize}{%

```

2.18.1 Vanilla T_EX

`\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).

```

532 \def\LTxcmds@temp#1{%
533     \long\def\ltx@ifempty##1{%
534         \romannumeral0%
535         \iffalse{\fi
536             \expandafter\ltx@gobble\expandafter{%

```

```

537         \expandafter{\string##1}%
538         \expandafter\ltx@gobble\string
539     }%
540     \expandafter\ltx@firstofthree\expandafter
541     {\iffalse}\fi
542     \expandafter#1\ltx@secondoftwo
543 }%
544 \expandafter#1\ltx@firstoftwo
545 }%

\ltx@ifblank

546 \long\def\ltx@ifblank##1{%
547     \romannumeral0%
548     \iffalse{\fi
549         \expandafter\expandafter\expandafter\ltx@gobble
550         \expandafter\expandafter\expandafter{%
551             \expandafter\expandafter\expandafter{%
552                 \expandafter\string\ltx@gobble##1.%
553             }%
554             \expandafter\ltx@gobble\string
555         }%
556         \expandafter\ltx@firstofthree\expandafter
557         {\iffalse}\fi
558         \expandafter#1\ltx@secondoftwo
559     }%
560     \expandafter#1\ltx@firstoftwo
561 }%
562 }%
563 \LTXcmds@temp{ }%

564 }{%

```

2.18.2 With \detokenize

Ahmed Musa provided `\ifstrempy` 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

565 \long\def\ltx@ifempty#1{%
566     \romannumeral%
567     \csname
568         LTXcmds@ifempty%
569         \ifcat$\detokenize{#1}$%
570         @%
571     \fi
572     \endcsname
573 }%

\LTXcmds@ifempty@

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

\LTXcmds@ifempty

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

```

2.18.3 \ltx@ifblank

```

\ltx@ifblank

576 \long\def\ltx@ifblank#1{%
577     \romannumeral%
578     \csname

```

```

579      LTXcmds@ifempty%
580      \ifcat$\detokenize\expandafter{\ltx@gobble#1.}$%
581      @%
582      \fi
583      \endcsname
584  }%

585 }

```

2.19 \ltx@zapspace

\ltx@zapspace

```

586 \long\def\ltx@zapspace#1{%
587   \romannumeral
588   \LTXcmds@zapspace\ltx@zero#1 \@nil
589 }

```

\LTXcmds@zapspace

```

590 \long\def\LTXcmds@zapspace#1 #2\@nil{%
591   \ltx@ifempty{#2}{%
592     #1%
593   }{%
594     \LTXcmds@zapspace#1#2\@nil
595   }%
596 }

```

2.20 \ltx@ifboxempty

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

```

597 \ltx@ifundefined{lastnodetype}{%
598   \catcode'\$=9 %
599   \catcode'\&=14 %
600 }{%
601   \catcode'\$=14 %
602   \catcode'\&=9 %
603 }

```

\ltx@ifboxempty

```

604 \def\ltx@ifboxempty#1{%
605   \ifvoid#1\relax
606   \expandafter\ltx@secondoftwo
607   \else

```

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

```

608 &   \begingroup
609 &     \setbox\ltx@zero=\ifhbox#1\hbox\else\vbox\fi{%
610 &       \ifhmode\unhcopy\else\unvcopy\fi#1\relax
611 &       \expandafter
612 &     }%
613 &   \expandafter\endgroup
614 &   \ifnum\lastnodetype<\ltx@zero
615 &     \expandafter\expandafter\expandafter\ltx@firstoftwo
616 &   \else
617 &     \expandafter\expandafter\expandafter\ltx@secondoftwo
618 &   \fi

```

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

```

619 $   \begingroup
620 $     \setbox\ltx@zero=\ifhbox#1\hbox\else\vbox\fi{%
621 $       \penalty\ltx@one
622 $       \ifhmode\unhcopy\else\unvcopy\fi#1\relax

```



```

623 $      \expandafter
624 $      }%
625 $      \ifnum\lastpenalty=\ltx@one
Box 0 has been changed and is restored by closing the group.
626 $      \endgroup
627 $      \beginingroup
628 $      \setbox\ltx@zero=\ifhbox#1\hbox\else\vbox\fi{%
629 $      \penalty\ltx@two
630 $      \ifhmode\unhcopy\else\unvcopy\fi#1\relax
631 $      \expandafter
632 $      }%
633 $      \ifnum\lastpenalty=\ltx@two
634 $      \def\next{\endgroup\expandafter\ltx@firstoftwo}%
635 $      \else
636 $      \def\next{\endgroup\expandafter\ltx@secondoftwo}%
637 $      \fi
638 $      \else
639 $      \def\next{\endgroup\expandafter\ltx@secondoftwo}%
640 $      \fi
641 $      \next
642 \fi
643 }

```

`\ltx@ifboxvoidoreempty`

```

644 \def\ltx@ifboxvoidoreempty#1{%
645   \ifvoid#1\relax
646     \expandafter\ltx@thirdoffour
647   \fi
648   \ltx@ifboxempty{#1}%
649 }

650 \LTXcmds@AtEnd%
651 \</package>

```

3 Test

3.1 Catcode checks for loading

```

652 <*test1>

653 \catcode'\{=1 %
654 \catcode'\}=2 %
655 \catcode'\#=6 %
656 \catcode'\@=11 %
657 \expandafter\ifx\csname count\endcsname\relax
658   \countdef\count@=255 %
659 \fi
660 \expandafter\ifx\csname @gobble\endcsname\relax
661   \long\def\@gobble#1{}%
662 \fi
663 \expandafter\ifx\csname @firstofone\endcsname\relax
664   \long\def\@firstofone#1{#1}%
665 \fi
666 \expandafter\ifx\csname loop\endcsname\relax
667   \expandafter\@firstofone
668 \else
669   \expandafter\@gobble
670 \fi
671 {%
672   \def\loop#1\repeat{%
673     \def\body{#1}%
674     \iterate

```

```

675 }%
676 \def\iterate{%
677   \body
678   \let\next\iterate
679   \else
680   \let\next\relax
681   \fi
682   \next
683 }%
684 \let\repeat=\fi
685 }%
686 \def\RestoreCatcodes{}
687 \count@=0 %
688 \loop
689   \edef\RestoreCatcodes{%
690     \RestoreCatcodes
691     \catcode\the\count@=\the\catcode\count@\relax
692   }%
693 \ifnum\count@<255 %
694   \advance\count@ 1 %
695 \repeat
696
697 \def\RangeCatcodeInvalid#1#2{%
698   \count@=#1\relax
699   \loop
700     \catcode\count@=15 %
701     \ifnum\count@<#2\relax
702       \advance\count@ 1 %
703     \repeat
704 }
705 \def\RangeCatcodeCheck#1#2#3{%
706   \count@=#1\relax
707   \loop
708     \ifnum#3=\catcode\count@
709     \else
710       \errmessage{%
711         Character \the\count@\space
712         with wrong catcode \the\catcode\count@\space
713         instead of \number#3%
714       }%
715     \fi
716     \ifnum\count@<#2\relax
717       \advance\count@ 1 %
718     \repeat
719 }
720 \def\space{ }
721 \expandafter\ifx\csname LoadCommand\endcsname\relax
722 \def\LoadCommand{\input ltxcmds.sty\relax}%
723 \fi
724 \def\Test{%
725   \RangeCatcodeInvalid{0}{47}%
726   \RangeCatcodeInvalid{58}{64}%
727   \RangeCatcodeInvalid{91}{96}%
728   \RangeCatcodeInvalid{123}{255}%
729   \catcode'\@=12 %
730   \catcode'\=0 %
731   \catcode'\%=14 %
732   \LoadCommand
733   \RangeCatcodeCheck{0}{36}{15}%
734   \RangeCatcodeCheck{37}{37}{14}%
735   \RangeCatcodeCheck{38}{47}{15}%
736   \RangeCatcodeCheck{48}{57}{12}%

```

```

737 \RangeCatcodeCheck{58}{63}{15}%
738 \RangeCatcodeCheck{64}{64}{12}%
739 \RangeCatcodeCheck{65}{90}{11}%
740 \RangeCatcodeCheck{91}{91}{15}%
741 \RangeCatcodeCheck{92}{92}{0}%
742 \RangeCatcodeCheck{93}{96}{15}%
743 \RangeCatcodeCheck{97}{122}{11}%
744 \RangeCatcodeCheck{123}{255}{15}%
745 \RestoreCatcodes
746 }
747 \Test
748 \csname @@end\endcsname
749 \end
750 </test1>

```

3.2 Test \ltx@GobbleNum

```

751 <*test-gobble>
752 \catcode'\{=1 %
753 \catcode'\}=2 %
754 \catcode'\#=6 %
755 \expandafter\ifx\csname RequirePackage\endcsname\relax
756 \input ltxcmds.sty\relax
757 \else
758 \RequirePackage{ltxcmds}[2011/04/18]%
759 \fi
760 \catcode'\@=11 %
761 \def\msg#{\immediate\write16}%
762 \msg{[Test \string\ltx@GobbleNum]}%
763 \long\def\Test#1=#2\\{%
764 \edef\StrA{\ltx@GobbleNum#1}%
765 \expandafter\expandafter\expandafter\def
766 \expandafter\expandafter\expandafter\StrAA
767 \expandafter\expandafter\expandafter{\ltx@GobbleNum#1}%
768 \edef\StrB{#2}%
769 \ifx\StrA\StrB
770 \ifx\StrAA\StrB
771 \msg{* ok.}%
772 \else
773 \msg{StrAA: \StrAA}%
774 \msg{StrB: \StrB}%
775 \errhelp{Test: #1=#2}%
776 \errmessage{Test (two expansions) failed}%
777 \fi
778 \else
779 \msg{StrA: \StrA}%
780 \msg{StrB: \StrB}%
781 \errhelp{Test: #1=#2}%
782 \errmessage{Test (edef) failed!}%
783 \fi
784 }
785 \Test0abc=abc\\
786 \Test1abc=bc\\
787 \Test2abc=c\\
788 \Test3abcd=d\\
789 \Test4abcde=e\\
790 \Test5abcdef=f\\
791 \Test6abcdefg=g\\
792 \Test7abcdefgh=h\\
793 \Test8abcdefghi=i\\
794 \Test9abcdefghij=j\\
795 \Test{10}0123456789X=X\\
796 \Test{12}abcdefghijkmlm=m\\

```

```

797 \Test{700}%
798 012345678901234567890123456789012345678901234567890123456789%
799 012345678901234567890123456789012345678901234567890123456789%
800 012345678901234567890123456789012345678901234567890123456789%
801 012345678901234567890123456789012345678901234567890123456789%
802 012345678901234567890123456789012345678901234567890123456789%
803 012345678901234567890123456789012345678901234567890123456789%
804 012345678901234567890123456789012345678901234567890123456789%
805 012345678901234567890123456789012345678901234567890123456789%
806 012345678901234567890123456789012345678901234567890123456789%
807 012345678901234567890123456789012345678901234567890123456789%
808 X=X\\
809 \Test{-1}abc=abc\\
810 \Test2\par\par\relax=\relax\\
811
812 \begingroup
813   \count1=2 %
814   \Test{\count1}abc=c\\%
815 \endgroup
816
817 \ltx@ifundefined{numexpr}{%
818 }{%
819   \Test{1+1}abc=c\\%
820 }
821
822 \msg{[Test \string\ltx@CdrNum]}%
823 \long\def\Test#1=#2\\{%
824   \edef\StrA{\ltx@CdrNum#1\@nil}%
825   \expandafter\expandafter\expandafter\def
826     \expandafter\expandafter\expandafter\StrAA
827     \expandafter\expandafter\expandafter{\ltx@CdrNum#1\@nil}%
828   \edef\StrB{#2}%
829   \ifx\StrA\StrB
830     \ifx\StrAA\StrB
831       \msg{* ok.}%
832     \else
833       \msg{StrAA: \meaning\StrAA}%
834       \msg{StrB: \meaning\StrB}%
835       \errhelp{Test: #1=#2}%
836       \errmessage{Test (two expansions) failed}%
837     \fi
838   \else
839     \msg{StrA: \StrA}%
840     \msg{StrB: \StrB}%
841     \errhelp{Test: #1=#2}%
842     \errmessage{Test (edef) failed!}%
843   \fi
844 }
845 \Test0abc=abc\\
846 \Test1abc=bc\\
847 \Test2abc=c\\
848 \Test3abcd=d\\
849 \Test4abcde=e\\
850 \Test5abcdef=f\\
851 \Test6abcdefg=g\\
852 \Test7abcdefgh=h\\
853 \Test8abcdefghi=i\\
854 \Test9abcdefghij=j\\
855 \Test{10}0123456789X=X\\
856 \Test{12}abcdefghijklm=m\\
857 \Test{700}%
858 012345678901234567890123456789012345678901234567890123456789%

```

```

859 0123456789012345678901234567890123456789012345678901234567890123456789%
860 0123456789012345678901234567890123456789012345678901234567890123456789%
861 0123456789012345678901234567890123456789012345678901234567890123456789%
862 0123456789012345678901234567890123456789012345678901234567890123456789%
863 0123456789012345678901234567890123456789012345678901234567890123456789%
864 0123456789012345678901234567890123456789012345678901234567890123456789%
865 0123456789012345678901234567890123456789012345678901234567890123456789%
866 0123456789012345678901234567890123456789012345678901234567890123456789%
867 0123456789012345678901234567890123456789012345678901234567890123456789%
868 X=X\\
869 \Test{-1}abc=abc\\
870 \Test2\par\par\relax=\relax\\
871
872 \msg{[Test \string\ltx@CarNum]}%
873 \long\def\Test#1=#2\\{%
874   \edef\StrA{\ltx@CarNum#1\@nil}%
875   \expandafter\expandafter\expandafter\def
876   \expandafter\expandafter\expandafter\StrAA
877   \expandafter\expandafter\expandafter{\ltx@CarNum#1\@nil}%
878   \edef\StrB{#2}%
879   \ifx\StrA\StrB
880     \ifx\StrAA\StrB
881       \msg{* ok.}%
882     \else
883       \msg{StrAA: \meaning\StrAA}%
884       \msg{StrB: \meaning\StrB}%
885       \errhelp{Test: #1=#2}%
886       \errmessage{Test (two expansions) failed}%
887     \fi
888   \else
889     \msg{StrA: \StrA}%
890     \msg{StrB: \StrB}%
891     \errhelp{Test: #1=#2}%
892     \errmessage{Test (edef) failed!}%
893   \fi
894 }
895 \Test0abc=\\
896 \Test1abc=a\\
897 \Test2abc=ab\\
898 \Test3abc=abc\\
899 \Test3abcd=abcd\\
900 \Test4abcde=abcde\\
901 \Test{10}0123456789X=0123456789\\
902 \Test{12}abcdefghijklm=abcdefghijkl\\
903 \Test{700}%
904 0123456789012345678901234567890123456789012345678901234567890123456789%
905 0123456789012345678901234567890123456789012345678901234567890123456789%
906 0123456789012345678901234567890123456789012345678901234567890123456789%
907 0123456789012345678901234567890123456789012345678901234567890123456789%
908 0123456789012345678901234567890123456789012345678901234567890123456789%
909 0123456789012345678901234567890123456789012345678901234567890123456789%
910 0123456789012345678901234567890123456789012345678901234567890123456789%
911 0123456789012345678901234567890123456789012345678901234567890123456789%
912 0123456789012345678901234567890123456789012345678901234567890123456789%
913 0123456789012345678901234567890123456789012345678901234567890123456789%
914 X=%
915 0123456789012345678901234567890123456789012345678901234567890123456789%
916 0123456789012345678901234567890123456789012345678901234567890123456789%
917 0123456789012345678901234567890123456789012345678901234567890123456789%
918 0123456789012345678901234567890123456789012345678901234567890123456789%
919 0123456789012345678901234567890123456789012345678901234567890123456789%
920 0123456789012345678901234567890123456789012345678901234567890123456789%

```

```

921 012345678901234567890123456789012345678901234567890123456789%
922 012345678901234567890123456789012345678901234567890123456789%
923 012345678901234567890123456789012345678901234567890123456789%
924 012345678901234567890123456789012345678901234567890123456789%
925 \\
926 \Test{-1}abc=\\
927 \Test2\par\par\relax=\par\par\\
928 \csname @@end\endcsname\end
929 </test-gobble>

```

3.3 Test \ltx@ifempty

```

930 <*test-ifempty>
931 \catcode'\{=1 %
932 \catcode'\}=2 %
933 \catcode'\#=6 %
934 \catcode'\@=11 %
935 \errorcontextlines=1000 %
936 \begingroup\expandafter\expandafter\expandafter\endgroup
937 \expandafter\ifx\csname RequirePackage\endcsname\relax
938   \input ltxcmds.sty\relax
939 \else
940   \RequirePackage{ltxcmds}[2011/04/18]%
941 \fi
942 \def\msg#\{\immediate\write16}
943 \def\TestY{\Y}
944 \def\TestN{\N}
945 \msg{* \string\ltx@ifempty}
946 \long\def\test#1{%
947   \begingroup
948     % Calculate expected test result via macro definition
949     \def\Stuff{#1}%
950     \ifx\Stuff\ltx@empty
951       \def\StuffEmpty{\Y}%
952     \else
953       \def\StuffEmpty{\N}%
954     \fi
955     % Test \ltx@ifempty
956     \expandafter\expandafter\expandafter\def
957     \expandafter\expandafter\expandafter\TestEmpty
958     \expandafter\expandafter\expandafter{%
959       \ltx@ifempty{#1}{\Y}{\N}%
960     }%
961     \ifx\StuffEmpty\TestEmpty
962       \msg{* Test OK}%
963     \else
964       \ltx@ifundefined{detokenize}{\}%
965       \msg{Stuff: [\detokenize{\Stuff}]}%
966     }%
967     \errmessage{Test failed!}%
968   \fi
969 \endgroup
970 }
971 \test{}
972 \test{a}
973 \test{abc}
974 \test{\par}
975 \test{ }
976 \test{\if}
977 \test{{\if}}
978 \test{\else}
979 \test{{\else}}
980 \test{\fi}

```

```

981 \test{ }\fi}
982 \test{ \or \ifcase}
983 \test{ } }
984 \test{ {a} }
985 \test{ { } abc}
986 \test{ { \par} }
987 \test{ { } \par}

988 \def \SpaceTwo#1{%
989   \def \SpaceTwo{#1#1}%
990 } \SpaceTwo{ }
991 \msg{* \string \ltx@ifblank}
992 \long \def \test#1{%
993   \begingroup
994     % Calculate expected test result via macro definition
995     \def \Stuff{#1}%
996     \ifx \Stuff \ltx@empty
997       \def \StuffEmpty{ \Y}%
998     \else
999       \ifx \Stuff \ltx@space
1000         \def \StuffEmpty{ \Y}%
1001       \else
1002         \ifx \Stuff \SpaceTwo
1003           \def \StuffEmpty{ \Y}%
1004         \else
1005           \def \StuffEmpty{ \N}%
1006         \fi
1007       \fi
1008     \fi
1009     % Test \ltx@ifblank
1010     \expandafter \expandafter \expandafter \def
1011     \expandafter \expandafter \expandafter \TestEmpty
1012     \expandafter \expandafter \expandafter {%
1013       \ltx@ifblank{#1}{ \Y}{ \N}%
1014     }%
1015     \ifx \StuffEmpty \TestEmpty
1016       \msg{* Test OK}%
1017     \else
1018       \ltx@ifundefined{detokenize}{ }{%
1019         \msg{Stuff: [ \detokenize{ \Stuff} ]}%
1020       }%
1021       \errmessage{Test failed!}%
1022     \fi
1023   \endgroup
1024 }
1025 \test{ }
1026 \test{a}
1027 \test{ \if}
1028 \test{ \else}
1029 \test{ \fi}
1030 \test{ \fi}
1031 \test{ \par}
1032 \test{ \par}
1033 \test{ { } }
1034 \test{ { } }
1035 \def \x#1{%
1036   \test{#1#1}%
1037   \test{#1#1{ } }%
1038   \test{#1#1 \par}%
1039   \test{#1#1 \else}%
1040 } \x{ }
1041 \csname @@end \endcsname \end
1042 \test-ifempty)

```

3.4 Test \ltx@zap@space

```

1043 <*test-zap@space>
1044 \catcode'\{=1 %
1045 \catcode'\}=2 %
1046 \catcode'\#=6 %
1047 \catcode'\@=11 %
1048 \errorcontextlines=1000 %
1049 \begingroup\expandafter\expandafter\expandafter\endgroup
1050 \expandafter\ifx\csname RequirePackage\endcsname\relax
1051 \input ltxcmds.sty\relax
1052 \else
1053 \RequirePackage{ltxcmds}[2011/04/18]%
1054 \fi
1055 \def\msg#{\immediate\write16}
1056 \def\space{ }
1057 \def\empty{}
1058 \msg{* \string\ltx@zap@space}
1059 \long\def\test#1#2{%
1060 \begingroup
1061 \def\TestInput{#1}%
1062 \def\TestExpected{#2}%
1063 % Test \ltx@zap@space
1064 \expandafter\expandafter\expandafter\def
1065 \expandafter\expandafter\expandafter\TestResult
1066 \expandafter\expandafter\expandafter{%
1067 \ltx@zap@space{#1}%
1068 }%
1069 \ifx\TestResult\TestExpected
1070 \msg{* Test OK}%
1071 \else
1072 \ltx@onelevel@sanitize\TestInput
1073 \ltx@onelevel@sanitize\TestExpected
1074 \ltx@onelevel@sanitize\TestResult
1075 \msg{* Input: \space\space\space[\TestInput]}%
1076 \msg{ \space Result: \space\space[\TestResult]}%
1077 \msg{ \space Expected: [\TestExpected]}%
1078 \errmessage{Test failed!}%
1079 \fi
1080 \endgroup
1081 }
1082 \long\def\etest#1#2{%
1083 \begingroup
1084 \edef\x{\endgroup
1085 \noexpand\test{#1}{#2}%
1086 }%
1087 \x
1088 }
1089 \catcode'\~=13 %
1090 \let~\noexpand
1091 \test{}{}
1092 \test{{}}{}
1093 \test{ {} }{}
1094 \test{{ } }{}
1095 \test{{} }{}
1096 \test{ {} }{}
1097 \test{ { } }{}
1098 \test{a {b} c}{a{b}c}
1099 \test{a bb ccc}{abbccc}
1100 \test{{a} {bb} {ccc}}{{a}{bb}{ccc}}
1101 \test{\par}{\par}
1102 \test{\if}{\if}
1103 \test{\space}{\space}

```



```

1104 \etest{\par\space\par}{\par\par}
1105 \etest{~\empty\space~\empty}{~\empty~\empty}
1106 \etest{~\fi\space~\else\space}{~\fi~\else}

1107 \csname @@end\endcsname\end
1108 </test-zapspace>

```

3.5 Test \ltx@ifboxempty

```

1109 <*test-ifboxempty>
1110 \catcode'\{=1 %
1111 \catcode'\}=2 %
1112 \catcode'\#=6 %
1113 \catcode'\@=11 %
1114 \begingroup\expandafter\expandafter\expandafter\endgroup
1115 \expandafter\ifx\csname RequirePackage\endcsname\relax
1116 \input ltxcmds.sty\relax
1117 \else
1118 \RequirePackage{ltxcmds}[2011/04/18]%
1119 \fi
1120 \def\msg#\{\immediate\write16}
1121 % make box 0 void
1122 \begingroup
1123 \setbox0=\box0 %
1124 \endgroup
1125 \ifvoid0 %
1126 \else
1127 \errmessage{Voiding box 0 failed}%
1128 \fi
1129 \setbox2=\box0 %
1130 \def\test#1#2{%
1131 \@test{#1}{#2}%
1132 \@@test{#1}{#2}%
1133 \chardef\x=#1%
1134 \@test\x{#2}%
1135 \@@test\x{#2}%
1136 }
1137 \def\@test#1#2{%
1138 \begingroup
1139 \setbox9=\hbox{%
1140 \def\TestExpected{#2}%
1141 \ltx@ifboxempty{#1}{%
1142 \def\TestResult{Y}%
1143 }{%
1144 \def\TestResult{N}%
1145 }%
1146 \ifx\TestExpected\TestResult
1147 \msg{* Test passed.}%
1148 \else
1149 \errmessage{Test failed!}%
1150 \fi
1151 }%
1152 \ifdim\wd9=0pt %
1153 \else
1154 \errmessage{Unwanted space?}%
1155 \fi
1156 \endgroup
1157 }
1158 \def\@@test#1#2{%
1159 \begingroup
1160 \setbox9=\hbox{%
1161 \def\TestExpected{#2}%
1162 \ifvoid#1\def\TestExpected{Y}\fi
1163 \ltx@ifboxvoidoreempty{#1}{%

```

```

1164         \def\TestResult{Y}%
1165     }{%
1166         \def\TestResult{N}%
1167     }%
1168     \ifx\TestExpected\TestResult
1169         \msg{* Test passed.}%
1170     \else
1171         \errmessage{Test failed!}%
1172     \fi
1173 }%
1174 \ifdim\wd9=0pt %
1175 \else
1176     \errmessage{Unwanted space?}%
1177 \fi
1178 \endgroup
1179 }
1180 \test0N
1181 \test2N
1182 \setbox0=\hbox{}
1183 \test0Y
1184 \setbox2=\hbox{}
1185 \test2Y
1186 \setbox0=\vbox{}
1187 \test0Y
1188 \setbox2=\vbox{}
1189 \test0Y
1190 \setbox0=\hbox{ }%
1191 \test0N
1192 \setbox2=\hbox{ }%
1193 \test2N
1194 \setbox0=\hbox{\penalty1}%
1195 \test0N
1196 \setbox2=\hbox{\penalty1}%
1197 \test2N
1198 \csname @@end\endcsname\end
1199 </test-ifboxempty>

```

3.6 Test for next character detection

```

1200 <*test-nextchar>
1201 \catcode'\{=1 %
1202 \catcode'\}=2 %
1203 \catcode'\#=6 %
1204 \catcode'\@=11 %
1205 \begingroup\expandafter\expandafter\expandafter\endgroup
1206 \expandafter\ifx\csname RequirePackage\endcsname\relax
1207     \input ltxcmds.sty\relax
1208     \input eolgrab.sty\relax
1209 \else
1210     \RequirePackage{ltxcmds}[2011/04/18]%
1211     \RequirePackage{eolgrab}[2011/01/12]%
1212 \fi
1213 \def\msg#{\immediate\write16}
1214 \begingroup
1215     \def\x#1{%
1216         \endgroup
1217         \let\TestSpaceToken= #1\relax
1218     }%
1219 \x{ }
1220 \def\TestSpace{ }
1221 \begingroup
1222     \lccode32=65 % space -> A
1223 \lowercase{%

```

```

1224 \endgroup
1225 \def\TestSpaceA{ }%
1226 }
1227 \def\TestCatch{%
1228   \eolgrab\@TestCatch
1229 }
1230 \def\@TestCatch#1{%
1231   \begingroup
1232     \def\x{#1}%
1233     \ifx\x\ltx@empty
1234       \else
1235         \ltx@onelevel@sanitize\x
1236         \errmessage{Unparsed stuff on line [\x]}%
1237       \fi
1238   \endgroup
1239 }
1240 \def\TestCmdM#1{%
1241   \TestCheckType{M}%
1242   \TestCatch
1243 }
1244 \def\TestCmdOM[#1]#2{%
1245   \TestCheckType{O}%
1246   \TestCatch
1247 }
1248 \def\TestCheckType#1{%
1249   \if\TestCmdType#1\relax
1250   \else
1251     \errmessage{Wrong type #1, expected: \TestCmdType}%
1252   \fi
1253 }
1254 \def\TestCmd#1{%
1255   \def\TestCmdType{#1}%
1256   \ltx@ifnextchar[\TestCmdOM\TestCmdM
1257 }
1258 \def\TestCmdExp#1{%
1259   \expandafter\TestCmd\expandafter#1%
1260 }
1261 \outer\def\TestOuter{
1262   \TestCmd O[o]{m}
1263   \TestCmd M{m}
1264   \TestCmd O [o]{m}
1265   \TestCmd M {m}
1266   \def\x#1{\def\x{#1#1}}\x{ }
1267   \TestCmdExp O\x[o]{m}
1268   \TestCmdExp M\x{m}
1269   \def\x#1{\def\x{#1#1#1}}\x{ }
1270   \TestCmdExp O\x[o]{m}
1271   \TestCmdExp M\x{m}
1272   \def\x{\TestSpaceToken}
1273   \TestCmdExp O\x[o]{m}
1274   \TestCmdExp M\x{m}
1275   \def\x{\TestSpaceToken\TestSpaceToken\TestSpaceToken}
1276   \TestCmdExp O\x[o]{m}
1277   \TestCmdExp M\x{m}
1278   \TestCmd M\TestSpace
1279   \TestOuter
1280   \TestCmd M \TestSpace
1281   \TestOuter
1282   \TestCmd M\iftrue
1283   \TestOuter
1284   \TestCmd M\iffalse
1285   \TestOuter

```

```

1286 \TestCmd M\else
1287 \TestOuter
1288 \TestCmd M\fi
1289 \TestOuter
1290 \TestCmd M \iftrue
1291 \TestOuter
1292 \TestCmd M \iffalse
1293 \TestOuter
1294 \TestCmd M \else
1295 \TestOuter
1296 %
1297 \def\TestCmd#1{%
1298   \def\TestCmdType{#1}%
1299   \ltx@ifnextchar@nospace[\TestCmdOM\TestCmdM
1300 }
1301 \TestCmd O[o]{m}
1302 \TestCmd M{m}
1303 \TestCmd M [
1304 \TestOuter
1305 \TestCmd M {m}
1306 \TestCmd M\iftrue
1307 \TestOuter
1308 \TestCmd M\iffalse
1309 \TestOuter
1310 \TestCmd M\else
1311 \TestCmd M\fi
1312 \TestOuter
1313 \TestOuter
1314 %
1315 \def\TestCmd#1{%
1316   \def\TestCmdType{#1}%
1317   \ltx@ifnextchar(\TestCmdPM\TestCmdM
1318 }
1319 \def\TestCmdPM(#1)#2{%
1320   \TestCheckType{P}%
1321   \TestCatch
1322 }
1323 \TestCmd P(p){m}
1324 \TestCmd M{m}
1325 \TestCmd P (p){m}
1326 \TestCmd M {m}
1327 %
1328 \def\TestCmd#1{%
1329   \def\TestCmdType{#1}%
1330   \ltx@ifnextchar{ }\TestCmdSM\TestCmdM
1331 }
1332 \def\TestCmdSM#1#{%
1333   \TestCheckType{S}%
1334   \begingroup
1335     \let\x= #1\relax
1336     \ifx\x\TestSpaceToken
1337       \else
1338         \errmessage{unexpected space token: \meaning#1}%
1339       \fi
1340   \endgroup
1341   \def\TestCmdType{M}%
1342   \TestCmdM
1343 }
1344 \TestCmd S {m}
1345 \TestCmd M{m}
1346 \def\x#1{\def\x{#1#1}}\x{ }
1347 \TestCmdExp S\x{m}

```

```

1348 %
1349 \def\TestCmd#1{%
1350   \def\TestCmdType{#1}%
1351   \ltx@ifnextchar\iffalse\TestCmdIM\TestCmdM
1352 }
1353 \def\TestCmdIM\iffalse#1{%
1354   \TestCheckType{I}%
1355   \TestCatch
1356 }
1357 \TestCmd M\iftrue
1358 \TestOuter
1359 \TestCmd M \iftrue
1360 \TestCmd I\iffalse\iffalse
1361 \TestCmd I \iffalse\iffalse
1362 \TestOuter
1363 %
1364 \def\TestCmd#1{%
1365   \def\TestCmdType{#1}%
1366   \ltx@ifnextchar@nospace\iffalse\TestCmdIM\TestCmdM
1367 }
1368 \TestCmd M\iftrue
1369 \TestOuter
1370 \TestCmd I\iffalse\iffalse
1371 \TestOuter
1372 \csname @@end\endcsname\end
1373 </test-nextchar>

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[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/
```

¹<http://ftp.ctan.org/tex-archive/>

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain $\mathrm{T}_{\mathrm{E}}\mathrm{X}$:

```
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):

<code>ltxcmds.sty</code>	\rightarrow <code>tex/generic/oberdiek/ltxcmds.sty</code>
<code>ltxcmds.pdf</code>	\rightarrow <code>doc/latex/oberdiek/ltxcmds.pdf</code>
<code>test/ltxcmds-test1.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ltxcmds-test1.tex</code>
<code>test/ltxcmds-test-gobble.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ltxcmds-test-gobble.tex</code>
<code>test/ltxcmds-test-ifempty.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ltxcmds-test-ifempty.tex</code>
<code>test/ltxcmds-test-zapspace.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ltxcmds-test-zapspace.tex</code>
<code>test/ltxcmds-test-ifboxempty.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ltxcmds-test-ifboxempty.tex</code>
<code>test/ltxcmds-test-nextchar.tex</code>	\rightarrow <code>doc/latex/oberdiek/test/ltxcmds-test-nextchar.tex</code>
<code>ltxcmds.dtx</code>	\rightarrow <code>source/latex/oberdiek/ltxcmds.dtx</code>

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`.

4.4 Refresh file name databases

If your $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ distribution (`te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , `mik $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` , ...) relies on file name databases, you must refresh these. For example, `te $\mathrm{T}_{\mathrm{E}}\mathrm{X}$` users run `texhash` or `mktextlsr`.

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 $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$. The `.dtx` chooses its action depending on the format:

plain $\mathrm{T}_{\mathrm{E}}\mathrm{X}$: Run `docstrip` and extract the files.

$\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$: Generate the documentation.

If you insist on using $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ for `docstrip` (really, `docstrip` does not need $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$), 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 `pdf $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$` :

```
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$1@news.albasani.net`, 2010-11-12.
<http://groups.google.com/group/comp.text.tex/msg/803bd57221a04996>
- [3] Ahmed Musa: *Re: TeX refuses to strip outer braces in argument*; newsgroup `comp.text.tex`, `news:f5496afe-40ed-42bd-b629-a2419ecf7c0d@o14g2000prn.googlegroups.com`, 2010-12-03.
<http://groups.google.com/group/comp.text.tex/msg/fbf7d61a0c3a807d>
- [4] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup `comp.text.tex`, `news:idbo94uka1@four.albasani.net`, 2010-12-03.
<http://groups.google.com/group/comp.text.tex/msg/0c230ee479487962>
- [5] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup `comp.text.tex`, `news:idbpu4cgl1@news.albasani.net`, 2010-12-03.
<http://groups.google.com/group/comp.text.tex/msg/bbef4263390d647b>
- [6] Ulrich Diez: *Re: TeX refuses to strip outer braces in argument*; newsgroup `comp.text.tex`, `news:idd4ga$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@fourthofffour` added.

[2011/02/05 v1.17]

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

[2011/03/16 v1.18]

- `\ltx@ifclasslater` fixed.

[2011/04/14 v1.19]

- `\ltx@ifnextchar`: detection of optional spaces modified.
- `\ltx(Loc,Glob)(Toks,Dimen,Skip)(A,B,C,D,E)` added.

[2011/04/18 v1.20]

- `\ltx@ifnextchar` with conditional support (thanks GL for bug report).

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.

Symbols	
<code>\#</code> . 230, 655, 754, 933, 1046, 1112, 1203	853, 854, 855, 856, 868, 869, 870, 873, 895, 896, 897, 898, 899, 900, 901, 902, 925, 926, 927
<code>\\$</code> 598, 601	<code>\{</code> . 235, 653, 752, 931, 1044, 1110, 1201
<code>\%</code> 220, 731	<code>\}</code> . 240, 654, 753, 932, 1045, 1111, 1202
<code>\&</code> 599, 602	<code>\~</code> 1089
<code>\.</code> 463	
<code>\@</code> . 656, 729, 760, 934, 1047, 1113, 1204	
<code>\@test</code> 1132, 1135, 1158	A
<code>\@TestCatch</code> 1228, 1230	<code>\advance</code> 694, 702, 717
<code>\@firstofone</code> 664, 667	<code>\aftergroup</code> 29
<code>\@gobble</code> 661, 669	B
<code>\@nil</code> . . 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 205, 214, 248, 253, 270, 276, 391, 393, 588, 590, 594, 824, 827, 874, 877	<code>\body</code> 673, 677
<code>\@test</code> 1131, 1134, 1137	<code>\box</code> 489, 500, 1123, 1129
<code>\@undefined</code> 58	C
<code>\</code> 225, 730, 763, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 808, 809, 810, 814, 819, 823, 845, 846, 847, 848, 849, 850, 851, 852,	<code>\catcode</code> 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 598, 599, 601, 602, 653, 654, 655,

656, 691, 700, 708, 712, 729, 730, 731, 752, 753, 754, 760, 931, 932, 933, 934, 1044, 1045, 1046, 1047, 1089, 1110, 1111, 1112, 1113, 1201, 1202, 1203, 1204	\chardef . 116, 117, 118, 119, 120, 1133	\iffalse 262, 285, 535, 541, 548, 557, 1284, 1292, 1308, 1351, 1353, 1360, 1361, 1366, 1370	\ifhbox 609, 620, 628
\count 813, 814	\count@ 658, 687, 691, 693, 694, 698, 700, 701, 702, 706, 708, 711, 712, 716, 717	\ifnum 367, 369, 371, 398, 614, 625, 633, 693, 701, 708, 716	\ifhmode 610, 622, 630
\countdef 658	\csname 14, 21, 50, 66, 76, 160, 165, 192, 197, 254, 256, 259, 261, 264, 277, 279, 282, 284, 287, 296, 298, 306, 319, 357, 452, 495, 567, 578, 657, 660, 663, 666, 721, 748, 755, 928, 937, 1041, 1050, 1107, 1115, 1198, 1206, 1372	\iftrue 257, 280, 1282, 1290, 1306, 1357, 1359, 1368	\ifvoid 496, 605, 645, 1125, 1162
\detokenize 569, 580, 965, 1019	\dimendef 134, 135, 136, 137, 138, 139, 140, 141, 142, 143	\ifix 15, 18, 21, 50, 58, 61, 296, 298, 306, 319, 413, 416, 427, 430, 448, 452, 475, 657, 660, 663, 666, 721, 755, 769, 770, 829, 830, 879, 880, 937, 950, 961, 996, 999, 1002, 1015, 1050, 1069, 1115, 1146, 1168, 1206, 1233, 1336	\immediate 23, 52, 761, 942, 1055, 1120, 1213
	D	\input 722, 756, 938, 1051, 1116, 1207, 1208	\iterate 674, 676, 678
	E		L
\empty 17, 18, 1057, 1105	\endcsname 14, 21, 50, 66, 76, 162, 168, 194, 200, 203, 254, 256, 259, 261, 264, 277, 279, 282, 284, 287, 296, 298, 306, 318, 319, 357, 452, 495, 572, 583, 657, 660, 663, 666, 721, 748, 755, 928, 937, 1041, 1050, 1107, 1115, 1198, 1206, 1372	\lastnodetype 614	\lastpenalty 625, 633
\end 749, 928, 1041, 1107, 1198, 1372	\endinginput 29, 115	\lccode 220, 225, 230, 235, 240, 1222	\leavevmode 509
\errhelp 775, 781, 835, 841, 885, 891	\errmessage 710, 776, 782, 836, 842, 886, 892, 967, 1021, 1078, 1127, 1149, 1154, 1171, 1176, 1236, 1251, 1338	\letLTxcmds@gttemp 431	\LoadCommand 722, 732
\errorcontextlines 935, 1048	\escapechar 246, 251, 268, 273	\loop 672, 688, 699, 707	\lowercase 221, 226, 231, 236, 241, 1223
\etest 1082, 1104, 1105, 1106		\ltx@(Loc, Glob) (Toks, Dimen, Skip) (A, B, C, D, E) 3	\ltx@active 119
	F		\ltx@backslashchar 224
\futurelet 445, 460, 471			\ltx@car 4, 180
	H		\ltx@carfour 4, 188
\hbox 487, 489, 498, 500, 516, 609, 620, 628, 1139, 1160, 1182, 1184, 1190, 1192, 1194, 1196			\ltx@CarNum 4, 190, 872, 874, 877
	I		\ltx@carthree 4, 186
\if 253, 276, 976, 977, 1027, 1102, 1249			\ltx@cartwo 4, 184
\ifcase 366, 982			\ltx@carzero 4, 182
\ifcat 569, 580			\ltx@cclv 120
\ifcsname 318			\ltx@cdr 181
\ifdim 1152, 1174			\ltx@cdrfour 189
			\ltx@CdrNum 209, 822, 824, 827
			\ltx@cdrthree 187
			\ltx@cdrtwo 185
			\ltx@cdrzero 183
			\ltx@clsextension 6, 340, 346, 384
			\ltx@empty 5, 217, 414, 417, 428, 431, 950, 996, 1233
			\ltx@firstoffour 176
			\ltx@firstofone 4, 170, 314, 465, 520, 526
			\ltx@firstofthree 173, 540, 556
			\ltx@firstoftwo 171, 299, 307, 320, 325, 343, 380, 544, 560, 615, 634
			\ltx@fourthoffour 179
			\ltx@GlobalAppendToMacro 7, 412
			\ltx@GlobDimenA 139
			\ltx@GlobDimenB 140

<code>\ltx@GlobDimenC</code>	141	<code>\ltx@newif</code>	5 , 244
<code>\ltx@GlobDimenD</code>	142	<code>\ltx@one</code>	117 , 122 , 621 , 625
<code>\ltx@GlobDimenE</code>	143	<code>\ltx@onelevel@sanitize</code>	6 , 334 , 1072 , 1073 , 1074 , 1235
<code>\ltx@GlobSkipA</code>	149	<code>\ltx@percentchar</code>	219
<code>\ltx@GlobSkipB</code>	150	<code>\ltx@pkgextension</code>	341 , 349 , 387
<code>\ltx@GlobSkipC</code>	151	<code>\ltx@RemovePrefix</code>	6 , 330 , 332 , 337
<code>\ltx@GlobSkipD</code>	152	<code>\ltx@ReturnAfterElseFi</code>	216
<code>\ltx@GlobSkipE</code>	153	<code>\ltx@ReturnAfterFi</code>	5 , 215
<code>\ltx@GlobToksA</code>	129	<code>\ltx@rightbracechar</code>	239
<code>\ltx@GlobToksB</code>	130	<code>\ltx@secondoffour</code>	177
<code>\ltx@GlobToksC</code>	131	<code>\ltx@secondofthree</code>	174
<code>\ltx@GlobToksD</code>	132	<code>\ltx@secondoftwo</code>	172 , 301 , 309 , 322 , 343 , 363 , 378 , 542 , 558 , 606 , 617 , 636 , 639
<code>\ltx@GlobToksE</code>	133	<code>\ltx@space</code>	5 , 218 , 377 , 999
<code>\ltx@gobble</code>	3 , 154 , 312 , 536 , 538 , 549 , 552 , 554 , 580	<code>\ltx@StripPrefix</code>	331 , 404 , 405 , 406
<code>\ltx@gobblefour</code>	157	<code>\ltx@thirdoffour</code>	178 , 646
<code>\ltx@GobbleNum</code>	3 , 158 , 212 , 762 , 764 , 767	<code>\ltx@thirdofthree</code>	175
<code>\ltx@gobblethree</code>	156	<code>\ltx@two</code>	118 , 629 , 633
<code>\ltx@gobbletwo</code>	155	<code>\ltx@undefined</code>	413 , 427
<code>\ltx@hashchar</code>	229	<code>\ltx@zapspace</code>	8 , 586 , 1058 , 1063 , 1067
<code>\ltx@ifblank</code>	8 , 546 , 576 , 991 , 1009 , 1013	<code>\ltx@zero</code>	3 , 116 , 206 , 487 , 489 , 491 , 588 , 609 , 614 , 620 , 628
<code>\ltx@ifBoxEmpty</code>	8 , 604 , 648 , 1141	<code>\LTxcmds@ifnextchar</code>	453 , 459
<code>\ltx@ifBoxVoidOrEmpty</code>	9 , 644 , 1163	<code>\LTxcmds@@ParseVersion</code>	391 , 393
<code>\ltx@ifclasslater</code>	7 , 383	<code>\LTxcmds@AtEnd</code>	95 , 96 , 115 , 650
<code>\ltx@ifclassloaded</code>	6 , 345	<code>\LTxcmds@CarNum</code>	193 , 196
<code>\ltx@ifempty</code>	8 , 532 , 565 , 591 , 945 , 955 , 959	<code>\LTxcmds@CarNumFinish</code>	205
<code>\ltx@iffilelater</code>	351 , 384 , 387	<code>\LTxcmds@CdrNum</code>	211 , 214
<code>\ltx@iffileloaded</code>	6 , 342 , 346 , 349 , 352	<code>\LTxcmds@CharToken</code>	442 , 448 , 468 , 475
<code>\ltx@ifnextchar</code>	7 , 440 , 1256 , 1317 , 1330 , 1351	<code>\LTxcmds@Cm</code>	199
<code>\ltx@ifnextchar@nospace</code>	7 , 466 , 1299 , 1366	<code>\LTxcmds@Cx</code>	202
<code>\ltx@ifpackagelater</code>	386	<code>\LTxcmds@Gm</code>	167
<code>\ltx@ifpackageloaded</code>	348	<code>\LTxcmds@GobbleNum</code>	161 , 164
<code>\ltx@ifUndefined</code>	5 , 304 , 328 , 389 , 481 , 482 , 483 , 484 , 518 , 531 , 597 , 817 , 964 , 1018	<code>\LTxcmds@gtemp</code>	426 , 427 , 428 , 430 , 435 , 436 , 438
<code>\ltx@ifundefined</code>	5 , 297 , 317 , 328 , 343	<code>\LTxcmds@ifempty</code>	575
<code>\ltx@leavevmode</code>	8 , 481 , 515	<code>\LTxcmds@ifempty@</code>	574
<code>\ltx@leftbracechar</code>	234	<code>\LTxcmds@ifLater</code>	353 , 365
<code>\ltx@LocalAppendToMacro</code>	425	<code>\LTxcmds@ifnextchar</code>	445 , 447 , 462
<code>\ltx@LocalExpandAfter</code>	6 , 289 , 295	<code>\LTxcmds@ifnextchar@nospace</code>	471 , 473
<code>\ltx@LocDimenA</code>	134	<code>\LTxcmds@LetToken</code>	445 , 448 , 461 , 471 , 475
<code>\ltx@LocDimenB</code>	135	<code>\LTxcmds@newglobalif</code>	270 , 272
<code>\ltx@LocDimenC</code>	136	<code>\LTxcmds@newif</code>	248 , 250
<code>\ltx@LocDimenD</code>	137	<code>\LTxcmds@num</code>	162 , 194 , 518
<code>\ltx@LocDimenE</code>	138	<code>\LTxcmds@ParseVersion</code>	355 , 361 , 390 , 397
<code>\ltx@LocSkipA</code>	144	<code>\LTxcmds@SpaceToken</code>	452 , 465
<code>\ltx@LocSkipB</code>	145	<code>\LTxcmds@temp</code>	532 , 563
<code>\ltx@LocSkipC</code>	146	<code>\LTxcmds@VoidBox</code>	495 , 496 , 498 , 500 , 503
<code>\ltx@LocSkipD</code>	147	<code>\LTxcmds@zapspace</code>	588 , 590
<code>\ltx@LocSkipE</code>	148		
<code>\ltx@LocToksA</code>	124 , 421 , 422 , 435 , 436 , 443 , 449 , 469 , 476		
<code>\ltx@LocToksB</code>	125 , 444 , 455 , 470 , 478		
<code>\ltx@LocToksC</code>	126		
<code>\ltx@LocToksD</code>	127		
<code>\ltx@LocToksE</code>	128		
<code>\ltx@mbox</code>	8 , 514		
<code>\ltx@minusone</code>	121		
<code>\ltx@newglobalif</code>	5 , 266		

M

<code>\meaning</code>	337 , 833 , 834 , 883 , 884 , 1338
<code>\msg</code>	761 , 762 , 771 , 773 , 774 , 779 , 780 , 822 , 831 , 833 , 834 , 839 , 840 , 872 , 881 , 883 , 884 , 889 , 890 , 942 , 945 , 962 , 965 , 991 , 1016 , 1019 , 1055 , 1058 , 1070 , 1075 , 1076 , 1077 , 1120 , 1147 , 1169 , 1213

N	
\N	944, 953, 959, 1005, 1013
\next	634, 636, 639, 641, 678, 680, 682
\number	354, 360, 521, 713
\numexpr	527
O	
\outer	1261
P	
\PackageInfo	26
\par	810, 870, 927, 974, 986, 987, 1031, 1032, 1038, 1101, 1104
\pdflastmatch	404, 405, 406
\pdfmatch	398
\penalty	621, 629, 1194, 1196
\ProvidesPackage	19, 67
Q	
\quitvmode	512
R	
\RangeCatcodeCheck	705, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744
\RangeCatcodeInvalid	697, 725, 726, 727, 728
\repeat	672, 684, 695, 703, 718
\RequirePackage	758, 940, 1053, 1118, 1210, 1211
\RestoreCatcodes	686, 689, 690, 745
\romannumeral	159, 162, 191, 194, 210, 463, 534, 547, 566, 577, 587
S	
\setbox	487, 489, 498, 500, 609, 620, 628, 1123, 1129, 1139, 1160, 1182, 1184, 1186, 1188, 1190, 1192, 1194, 1196
\skipdef	144, 145, 146, 147, 148, 149, 150, 151, 152, 153
\space	711, 712, 720, 1056, 1075, 1076, 1077, 1103, 1104, 1105, 1106
\SpaceTwo	988, 989, 990, 1002
\StrA	764, 769, 779, 824, 829, 839, 874, 879, 889
\StrAA	766, 770, 773, 826, 830, 833, 876, 880, 883
\StrB	768, 769, 770, 774, 780, 828, 829, 830, 834, 840, 878, 879, 880, 884, 890
\Stuff	949, 950, 965, 995, 996, 999, 1002, 1019
\StuffEmpty	951, 953, 961, 997, 1000, 1003, 1005, 1015
T	
\Test	724, 747, 763, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 809, 810, 814, 819, 823, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 869, 870, 873, 895, 896, 897, 898, 899, 900, 901, 902, 903, 926, 927
\test	946, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 992, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1036, 1037, 1038, 1039, 1059, 1085, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1130, 1180, 1181, 1183, 1185, 1187, 1189, 1191, 1193, 1195, 1197
\TestCatch	1227, 1242, 1246, 1321, 1355
\TestCheckType	1241, 1245, 1248, 1320, 1333, 1354
\TestCmd	1254, 1259, 1262, 1263, 1264, 1265, 1278, 1280, 1282, 1284, 1286, 1288, 1290, 1292, 1294, 1297, 1301, 1302, 1303, 1305, 1306, 1308, 1310, 1311, 1315, 1323, 1324, 1325, 1326, 1328, 1344, 1345, 1349, 1357, 1359, 1360, 1361, 1364, 1368, 1370
\TestCmdExp	1258, 1267, 1268, 1270, 1271, 1273, 1274, 1276, 1277, 1347
\TestCmdIM	1351, 1353, 1366
\TestCmdM	1240, 1256, 1299, 1317, 1330, 1342, 1351, 1366
\TestCmdOM	1244, 1256, 1299
\TestCmdPM	1317, 1319
\TestCmdSM	1330, 1332
\TestCmdType	1249, 1251, 1255, 1298, 1316, 1329, 1341, 1350, 1365
\TestEmpty	957, 961, 1011, 1015
\TestExpected	1062, 1069, 1073, 1077, 1140, 1146, 1161, 1162, 1168
\TestInput	1061, 1072, 1075
\TestN	944
\TestOuter	1261, 1279, 1281, 1283, 1285, 1287, 1289, 1291, 1293, 1295, 1304, 1307, 1309, 1312, 1313, 1358, 1362, 1369, 1371
\TestResult	1065, 1069, 1074, 1076, 1142, 1144, 1146, 1164, 1166, 1168
\TestSpace	1220, 1278, 1280
\TestSpaceA	1225
\TestSpaceToken	1217, 1272, 1275, 1336
\TestY	943
\the	77, 78, 79, 80, 81, 82, 83, 84, 97, 422, 436, 449, 455, 474, 527, 691, 711, 712
\TMP@EnsureCode	94, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114
\toksdef	124, 125, 126, 127, 128, 129, 130, 131, 132, 133
U	
\unhbox	491, 503, 506
\unhcopy	610, 622, 630
\unvcopy	610, 622, 630

V		
\ vbox	609, 620, 628, 1186, 1188	56, 66, 75, 87, 1035, 1040, 1084, 1087, 1133, 1134, 1135, 1215,
\ voidb@x	506	1219, 1232, 1233, 1235, 1236, 1266, 1267, 1268, 1269, 1270,
W		
\ wd	1152, 1174	1271, 1272, 1273, 1274, 1275, 1276, 1277, 1335, 1336, 1346, 1347
\ write	23, 52, 761, 942, 1055, 1120, 1213	
X		Y
\ x	14, 15, 18, 22, 26, 28, 51,	\ Y . 943, 951, 959, 997, 1000, 1003, 1013