

# The ltxcmds package

Heiko Oberdiek  
<heiko.oberdiek at googlemail.com>

2010/03/09 v1.4

## Abstract

The package ltxcmds exports some utility macros from the L<sup>A</sup>T<sub>E</sub>X kernel into a separate namespace and also provides them for other formats such as plain-T<sub>E</sub>X.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Numbers . . . . .	2
1.3	Argument killers . . . . .	2
1.4	Argument grabbers . . . . .	2
1.5	List helpers . . . . .	3
1.6	Tail recursion . . . . .	3
1.7	Empty macro . . . . .	3
1.8	Characters . . . . .	3
1.9	Boolean switch . . . . .	3
1.10	Command definitions . . . . .	3
1.11	Stripping . . . . .	4
1.12	File management . . . . .	4
	1.12.1 File extensions . . . . .	4
	1.12.2 Load check . . . . .	4
	1.12.3 Version date check . . . . .	4
1.13	Macro additions . . . . .	5
1.14	Macro \ltx@ifnextchar . . . . .	5
<b>2</b>	<b>Implementation</b>	<b>5</b>
2.1	Identification . . . . .	5
2.2	Numbers . . . . .	7
2.3	Argument killers . . . . .	7
2.4	Argument grabbers . . . . .	7
2.5	List helpers . . . . .	7
2.6	Tail recursion . . . . .	8
2.7	Empty macro . . . . .	8
2.8	Characters . . . . .	8
2.9	Boolean switch . . . . .	8
2.10	Command definitions . . . . .	9
2.11	Stripping . . . . .	9
2.12	File management . . . . .	10
	2.12.1 File extensions . . . . .	10
	2.12.2 Load check . . . . .	10
	2.12.3 Version date check . . . . .	10
2.13	Macro additions . . . . .	11
2.14	Macro \ltx@ifnextchar . . . . .	12

<b>3</b>	<b>Test</b>	<b>13</b>
3.1	Catcode checks for loading . . . . .	13
<b>4</b>	<b>Installation</b>	<b>14</b>
4.1	Download . . . . .	14
4.2	Bundle installation . . . . .	14
4.3	Package installation . . . . .	15
4.4	Refresh file name databases . . . . .	15
4.5	Some details for the interested . . . . .	15
<b>5</b>	<b>History</b>	<b>16</b>
	[2009/08/05 v1.0] . . . . .	16
	[2009/12/12 v1.1] . . . . .	16
	[2010/01/28 v1.2] . . . . .	16
	[2010/03/01 v1.3] . . . . .	16
	[2010/03/09 v1.4] . . . . .	16
<b>6</b>	<b>Index</b>	<b>16</b>

# 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 L<sup>A</sup>T<sub>E</sub>X'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

<code>\ltx@zero</code>
<code>\ltx@one</code>
<code>\ltx@two</code>
<code>\ltx@cclv</code>

These commands are numbers 0, 1, 2 and 255. They are not digits and a space is not gobbled afterwards.

## 1.3 Argument killers

<code>\ltx@gobble {⟨1⟩}</code>	→
<code>\ltx@gobbletwo {⟨1⟩} {⟨2⟩}</code>	→
<code>\ltx@gobblethree {⟨1⟩} {⟨2⟩} {⟨3⟩}</code>	→
<code>\ltx@gobblefour {⟨1⟩} {⟨2⟩} {⟨3⟩} {⟨4⟩}</code>	→

## 1.4 Argument grabbers

<code>\ltx@firstofone {⟨1⟩}</code>	→	<code>⟨1⟩</code>
<code>\ltx@firstoftwo {⟨1⟩} {⟨2⟩}</code>	→	<code>⟨1⟩</code>
<code>\ltx@secondoftwo {⟨1⟩} {⟨2⟩}</code>	→	<code>⟨2⟩</code>

## 1.5 List helpers

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

## 1.6 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>

## 1.7 Empty macro

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

## 1.8 Characters

<code>\ltx@space</code>
<code>\ltx@percentchar</code>
<code>\ltx@backslashchar</code>

## 1.9 Boolean switch

<code>\ltx@newif {⟨cmd⟩}</code>
---------------------------------

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

## 1.10 Command definitions

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

If  $\varepsilon\text{-}\text{\TeX}$  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”.

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

If  $\varepsilon\text{-}\text{\TeX}$  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  $\varepsilon\text{-}\text{\TeX}$ .

<code>\ltx@LocalExpandAfter</code>
------------------------------------

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

<code>\ltx@RemovePrefix</code> <code>\ltx@StripPrefix</code>
---

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<sup>A</sup>T<sub>E</sub>X’s `\strip@prefix`, whereas macro `\ltx@StripPrefix` expands the next token once before stripping the prefix.

## 1.12 File management

All macros in this section are expandable like the counterparts of the L<sup>A</sup>T<sub>E</sub>X kernel. Also they can be used after the preamble.

### 1.12.1 File extensions

<code>\ltx@clsextension</code> <code>\ltx@pkgextension</code>
--

If `\@clsextension/\@pkgextension` exists then `\ltx@clsextension/\ltx@pkgextension` returns this macro, otherwise the result is `cls/sty`.

### 1.12.2 Load check

<code>\ltx@ifclassloaded {&lt;class&gt;} {&lt;yes&gt;} {&lt;no&gt;}</code> <code>\ltx@ifpackageloaded {&lt;package&gt;} {&lt;yes&gt;} {&lt;no&gt;}</code>
--

If the `<class>/<package>` are loaded the macros `\ltx@ifclassloaded/\ltx@ifpackageloaded` call the `<yes>` argument. Otherwise `<no>` is executed. Both `<class>` and `<package>` are specified without extension.

<code>\ltx@iffileloaded {&lt;file&gt;} {&lt;yes&gt;} {&lt;no&gt;}</code>
--

If L<sup>A</sup>T<sub>E</sub>X’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<sup>A</sup>T<sub>E</sub>X 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 T<sub>E</sub>X’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

<code>\ltx@ifclasslater {&lt;class&gt;} {&lt;date&gt;} {&lt;yes&gt;} {&lt;no&gt;}</code> <code>\ltx@ifpackagelater {&lt;package&gt;} {&lt;date&gt;} {&lt;yes&gt;} {&lt;no&gt;}</code> <code>\ltx@iffilelater {&lt;file&gt;} {&lt;date&gt;} {&lt;yes&gt;} {&lt;no&gt;}</code>
--

If a `\ProvidesClass/\ProvidesPackage/\ProvidesFile` command with exact the same class/package/file was executed before with an optional argument that starts with a L<sup>A</sup>T<sub>E</sub>X version date, then this version date is compared with the

argument  $\langle date \rangle$ . If they are equal or if the version date is the later date, then  $\langle yes \rangle$  is called. In all other cases  $\langle no \rangle$  is executed.

A L<sup>A</sup>T<sub>E</sub>X 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 pdfT<sub>E</sub>X'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 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> in 1994.

### 1.13 Macro additions

```
\ltx@GlobalAppendToMacro { $\langle cmd \rangle$ } { $\langle addition \rangle$ }
\ltx@LocalAppendToMacro { $\langle cmd \rangle$ } { $\langle addition \rangle$ }
```

The  $\langle addition \rangle$  is appended to the parameterless macro  $\langle cmd \rangle$ . If  $\langle cmd \rangle$  is undefined or has the meaning `\relax`, then it will be initialized as empty macro before.

### 1.14 Macro `\ltx@ifnextchar`

```
\ltx@ifnextchar { $\langle char \rangle$ } { $\langle yes \rangle$ } { $\langle no \rangle$ }
```

If next character is  $\langle char \rangle$  then  $\langle yes \rangle$  is called, otherwise  $\langle no \rangle$ . The character is not removed.

## 2 Implementation

### 2.1 Identification

```
1 (*package)
```

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

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

Package identification:

```
30 \begingroup
31   \catcode35 6 % #
32   \catcode40 12 % (
33   \catcode41 12 % )
34   \catcode44 12 % ,
35   \catcode45 12 % -
36   \catcode46 12 % .
37   \catcode47 12 % /
38   \catcode58 12 % :
39   \catcode64 11 % @
40   \catcode91 12 % [
41   \catcode93 12 % ]
42   \catcode123 1 % {
43   \catcode125 2 % }
44   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
45     \def\x#1#2#3[#4]{\endgroup
46       \immediate\write-1{Package: #3 #4}%
47       \xdef#1{#4}%
48     }%
49   \else
50     \def\x#1#2[#3]{\endgroup
51       #2[#3]}%
52     \ifx#1\@undefined
53       \xdef#1{#3}%
54     \fi
55     \ifx#1\relax
56       \xdef#1{#3}%
57     \fi
58   }%
59 \fi
60 \expandafter\x\csname ver@ltxcmds.sty\endcsname
61 \ProvidesPackage{ltxcmds}%
62 [2010/03/09 v1.4 LaTeX kernel commands for general use (H0)]
63 \begingroup
64   \catcode123 1 % {
65   \catcode125 2 % }
66   \def\x{\endgroup
67     \expandafter\edef\csname LTXcmds@AtEnd\endcsname{%
68       \catcode35 \the\catcode35\relax
69       \catcode64 \the\catcode64\relax
70       \catcode123 \the\catcode123\relax
71       \catcode125 \the\catcode125\relax
72     }%
73   }%
74 \x
75 \catcode35 6 % #
76 \catcode64 11 % @
77 \catcode123 1 % {
78 \catcode125 2 % }
79 \def\TMP@EnsureCode#1#2{%
80   \edef\LTXcmds@AtEnd{%
81     \LTXcmds@AtEnd
82     \catcode#1 \the\catcode#1\relax
83   }%
84   \catcode#1 #2\relax
85 }
86 \TMP@EnsureCode{40}{12}% (
87 \TMP@EnsureCode{41}{12}% )
88 \TMP@EnsureCode{45}{12}% -
89 \TMP@EnsureCode{46}{12}% .
90 \TMP@EnsureCode{47}{12}% /
```

```

91 \TMP@EnsureCode{60}{12}% <
92 \TMP@EnsureCode{61}{12}% =
93 \TMP@EnsureCode{62}{12}% >
94 \TMP@EnsureCode{91}{12}% [
95 \TMP@EnsureCode{96}{12}% ‘
96 \TMP@EnsureCode{93}{12}% ]
97 \TMP@EnsureCode{94}{12}% ^ (superscript) (!)
98 \TMP@EnsureCode{124}{12}% |

```

## 2.2 Numbers

```

\ltx@zero

99 \chardef\ltx@zero=0 %

```

```

\ltx@one

100 \chardef\ltx@one=1 %

```

```

\ltx@two

101 \chardef\ltx@two=2 %

```

```

\ltx@cclv

102 \chardef\ltx@cclv=255 %

```

## 2.3 Argument killers

```

\ltx@gobble

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

```

```

\ltx@gobbletwo

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

```

```

\ltx@gobblethree

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

```

```

\ltx@gobblefour

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

```

## 2.4 Argument grabbers

```

\ltx@firstofone

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

```

```

\ltx@firstoftwo

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

```

```

\ltx@secondoftwo

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

```

## 2.5 List helpers

```

\ltx@car

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

```

```

\ltx@cdr

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

```

## 2.6 Tail recursion

`\ltx@ReturnAfterFi`

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

`\ltx@ReturnAfterElseFi`

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

## 2.7 Empty macro

`\ltx@empty`

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

## 2.8 Characters

`\ltx@space`

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

`\ltx@percentchar`

```
116 \begingroup
117 \lccode'0='%\relax
118 \lowercase{\endgroup
119 \def\ltx@percentchar{0}%
120 }
```

`\ltx@backslashchar`

```
121 \begingroup
122 \lccode'0='\\relax
123 \lowercase{\endgroup
124 \def\ltx@backslashchar{0}%
125 }
```

## 2.9 Boolean switch

`\ltx@newif`

```
126 \def\ltx@newif#1{%
127 \begingroup
128 \escapechar=-1 %
129 \expandafter\endgroup
130 \expandafter\LTxcmds@newif\string#1\@nil
131 }
```

`\LTxcmds@newif`

```
132 \begingroup
133 \escapechar=-1 %
134 \expandafter\endgroup
135 \expandafter\def\expandafter\LTxcmds@newif\string\if#1\@nil{%
136 \expandafter\edef\csname#1true\endcsname{%
137 \let
138 \expandafter\noexpand\csname if#1\endcsname
139 \noexpand\iftrue
140 }%
141 \expandafter\edef\csname#1false\endcsname{%
142 \let
143 \expandafter\noexpand\csname if#1\endcsname
144 \noexpand\iffalse
145 }%
146 \csname#1false\endcsname
147 }
```



## 2.10 Command definitions

`\ltx@LocalExpandAfter`

```
148 \def\ltx@LocalExpandAfter{%
149   \begingroup
150   \expandafter\expandafter\expandafter
151   \endgroup
152   \expandafter
153 }

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

`\ltx@ifundefined`

```
156 \def\ltx@ifundefined#1{%
157   \expandafter\ifx\csname #1\endcsname\relax
158   \expandafter\ltx@firstoftwo
159   \else
160   \expandafter\ltx@secondoftwo
161   \fi
162 }%
```

`\ltx@ifUndefined`

```
163 \def\ltx@ifUndefined#1{%
164   \begingroup\expandafter\expandafter\expandafter\endgroup
165   \expandafter\ifx\csname #1\endcsname\relax
166   \expandafter\ltx@firstoftwo
167   \else
168   \expandafter\ltx@secondoftwo
169   \fi
170 }%

171 \expandafter\ltx@gobble
172 \else
173 \expandafter\ltx@firstofone
174 \fi
175 {%
```

`\ltx@ifundefined`

```
176 \def\ltx@ifundefined#1{%
177   \ifcsname #1\endcsname
178   \expandafter\ifx\csname #1\endcsname\relax
179   \expandafter\expandafter\expandafter\ltx@firstoftwo
180   \else
181   \expandafter\expandafter\expandafter\ltx@secondoftwo
182   \fi
183   \else
184   \expandafter\ltx@firstoftwo
185   \fi
186 }%
```

`\ltx@ifUndefined`

```
187 \let\ltx@ifUndefined\ltx@ifundefined
188 }
```

## 2.11 Stripping

`\ltx@RemovePrefix`

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

```

\ltx@StripPrefix
190 \def\ltx@StripPrefix{%
191   \expandafter\ltx@RemovePrefix
192 }

```

## 2.12 File management

### 2.12.1 File extensions

```

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

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

```

### 2.12.2 Load check

```

\ltx@iffileloaded
195 \def\ltx@iffileloaded#1{%
196   \ltx@ifundefined{ver@#1}\ltx@secondoftwo\ltx@firstoftwo
197 }

\ltx@ifclassloaded
198 \def\ltx@ifclassloaded#1{%
199   \ltx@iffileloaded{#1.\ltx@clsextension}%
200 }

\ltx@ifpackageloaded
201 \def\ltx@ifpackageloaded#1{%
202   \ltx@iffileloaded{#1.\ltx@pkgextension}%
203 }

```

### 2.12.3 Version date check

```

\ltx@iffilelater
204 \def\ltx@iffilelater#1#2{%
205   \ltx@iffileloaded{#1}{%
206     \expandafter\LTXcmds@IfLater\expandafter{%
207       \number
208       \expandafter\expandafter\expandafter\LTXcmds@ParseVersion
209       \expandafter\expandafter\expandafter{%
210         \csname ver@#1\endcsname
211       }%
212     \expandafter}\expandafter{%
213       \number
214       \expandafter\LTXcmds@ParseVersion\expandafter{#2}%
215     }%
216   }\ltx@secondoftwo
217 }

\LTXcmds@IfLater
218 \def\LTXcmds@IfLater#1#2{%
219   \ifcase 0%
220     \ifnum#1<19940101 %
221     \else
222       \ifnum#2<19940101 %
223       \else
224         \ifnum#2>#1 %
225         \else
226           1%

```

```

227         \fi
228     \fi
229     \fi
230     \ltx@space
231     \expandafter\ltx@secondoftwo
232 \else
233     \expandafter\ltx@firstoftwo
234 \fi
235 }

\ltx@ifclasslater
236 \def\ltx@ifclasslater#1{%
237     \ltx@ifclasslater{#1.\ltx@clsextension}%
238 }

\ltx@ifpackagelater
239 \def\ltx@ifpackagelater#1{%
240     \ltx@iffilelater{#1.\ltx@pkgextension}%
241 }

242 \ltx@ifundefined{pdfmatch}{%

\LTxcmds@ParseVersion
243     \def\LTxcmds@ParseVersion#1{%
244         \LTxcmds@@ParseVersion#10000/00/00\@nil
245     }%

\LTxcmds@@ParseVersion
246     \def\LTxcmds@@ParseVersion#1#2#3#4/#5#6/#7#8#9\@nil{%
247         #1#2#3#4#5#6#7#8%
248     }%

249 }{%

\LTxcmds@ParseVersion
250     \def\LTxcmds@ParseVersion#1{%
251         \ifnum\pdfmatch{%
252             ~%
253             (199[4-9] | [2-9] [0-9] [0-9] [0-9])/%
254             (0[1-9] | 1[0-2])/%
255             (0[1-9] | [1-2] [0-9] | 3[0-1])%
256         }{#1}=1 %
257         \ltx@StripPrefix\pdfastmatch1 %
258         \ltx@StripPrefix\pdfastmatch2 %
259         \ltx@StripPrefix\pdfastmatch3 %
260     \else
261         0%
262     \fi
263 }%
264 }

```

## 2.13 Macro additions

```

\ltx@GlobalAppendToMacro
265 \def\ltx@GlobalAppendToMacro#1#2{%
266     \ifx\ltx@undefined#1%
267         \let#1\ltx@empty
268     \else
269         \ifx\relax#1%
270             \let#1\ltx@empty
271         \fi

```

```

272 \fi
273 \begingroup
274   \toks0\expandafter{#1#2}%
275   \xdef#1{\the\toks0}%
276 \endgroup
277 }

```

`\ltx@LocalAppendToMacro`

```

278 \def\ltx@LocalAppendToMacro#1#2{%
279   \global\let\LTXcmds@gtemp#1%
280   \ifx\ltx@undefined\LTXcmds@gtemp
281     \global\let\LTXcmds@gtemp\ltx@empty
282   \else
283     \ifx\relax\LTXcmds@gtemp
284       \global\let\LTXcmds@gtemp\ltx@empty
285     \fi
286   \fi
287   \begingroup
288     \toks0\expandafter{\LTXcmds@gtemp#2}%
289     \xdef\LTXcmds@gtemp{\the\toks0}%
290   \endgroup
291   \let#1\LTXcmds@gtemp
292 }

```

## 2.14 Macro `\ltx@ifnextchar`

`\ltx@ifnextchar`

```

293 \long\def\ltx@ifnextchar#1#2#3{%
294   \begingroup
295   \let\LTXcmds@CharToken= #1\relax
296   \toks\ltx@zero{#2}%
297   \toks\ltx@two{#3}%
298   \futurelet\LTXcmds@LetToken\LTXcmds@ifnextchar
299 }

```

`\LTXcmds@ifnextchar`

```

300 \def\LTXcmds@ifnextchar{%
301   \ifx\LTXcmds@LetToken\LTXcmds@CharToken
302     \expandafter\endgroup\the\toks\expandafter\ltx@zero
303   \else
304     \ifx\LTXcmds@LetToken\LTXcmds@SpaceToken
305       \expandafter\expandafter\expandafter\LTXcmds@@ifnextchar
306     \else
307       \expandafter\endgroup\the\toks
308       \expandafter\expandafter\expandafter\ltx@two
309     \fi
310   \fi
311 }

```

`\LTXcmds@@ifnextchar`

```

312 \begingroup
313   \def\x#1{\endgroup
314     \def\LTXcmds@@ifnextchar#1{%
315       \futurelet\LTXcmds@LetToken\LTXcmds@ifnextchar
316     }%
317   }%
318   \x{ }

```

`\LTXcmds@SpaceToken`

```

319 \begingroup
320   \def\x#1{\endgroup

```

```

321 \let\LTxcmds@SpaceToken= #1%
322 }%
323 \x{ }

324 \LTxcmds@AtEnd
325 \endpackage

```

## 3 Test

### 3.1 Catcode checks for loading

```

326 \test1\

327 \catcode'\{=1 %
328 \catcode'\}=2 %
329 \catcode'\#=6 %
330 \catcode'\@=11 %
331 \expandafter\ifx\csname count@\endcsname\relax
332 \countdef\count@=255 %
333 \fi
334 \expandafter\ifx\csname @gobble\endcsname\relax
335 \long\def\@gobble#1{}%
336 \fi
337 \expandafter\ifx\csname @firstofone\endcsname\relax
338 \long\def\@firstofone#1{#1}%
339 \fi
340 \expandafter\ifx\csname loop\endcsname\relax
341 \expandafter\@firstofone
342 \else
343 \expandafter\@gobble
344 \fi
345 {%
346 \def\loop#1\repeat{%
347 \def\body{#1}%
348 \iterate
349 }%
350 \def\iterate{%
351 \body
352 \let\next\iterate
353 \else
354 \let\next\relax
355 \fi
356 \next
357 }%
358 \let\repeat=\fi
359 }%
360 \def\RestoreCatcodes{}
361 \count@=0 %
362 \loop
363 \edef\RestoreCatcodes{%
364 \RestoreCatcodes
365 \catcode\the\count@=\the\catcode\count@\relax
366 }%
367 \ifnum\count@<255 %
368 \advance\count@ 1 %
369 \repeat
370
371 \def\RangeCatcodeInvalid#1#2{%
372 \count@=#1\relax
373 \loop
374 \catcode\count@=15 %
375 \ifnum\count@<#2\relax
376 \advance\count@ 1 %

```

```

377 \repeat
378 }
379 \expandafter\ifx\csname LoadCommand\endcsname\relax
380 \def\LoadCommand{\input ltxcmds.sty\relax}%
381 \fi
382 \def\Test{%
383 \RangeCatcodeInvalid{0}{47}%
384 \RangeCatcodeInvalid{58}{64}%
385 \RangeCatcodeInvalid{91}{96}%
386 \RangeCatcodeInvalid{123}{255}%
387 \catcode'\@=12 %
388 \catcode'\=0 %
389 \catcode'\{=1 %
390 \catcode'\}=2 %
391 \catcode'\#=6 %
392 \catcode'\[=12 %
393 \catcode'\]=12 %
394 \catcode'\%=14 %
395 \catcode'\ =10 %
396 \catcode13=5 %
397 \LoadCommand
398 \RestoreCatcodes
399 }
400 \Test
401 \csname @@end\endcsname
402 \end
403 </test1>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[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 T<sub>E</sub>X 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/
```

---

<sup>1</sup>[ftp://ftp.ctan.org/tex-archive/](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):

```
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
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`.

### 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 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@GlobalAppendMacro`, `\ltx@LocalAppendMacro` 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`.

## 6 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; numbers in roman refer to the code lines where the entry is used.

Symbols		B	
<code>\#</code>	329, 391	<code>\body</code>	347, 351
<code>\%</code>	117, 394	<b>C</b>	
<code>\@</code>	330, 387	<code>\catcode</code>	3, 4, 5, 6, 7, 8, 9, 17, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 64, 65, 68, 69, 70, 71, 75, 76, 77, 78, 82, 84, 327, 328, 329, 330, 365, 374, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396
<code>\@firstofone</code>	338, 341	<code>\chardef</code>	99, 100, 101, 102
<code>\@gobble</code>	335, 343	<code>\count@</code>	332, 361, 365, 367, 368, 372, 374, 375, 376
<code>\@nil</code>	110, 111, 130, 135, 244, 246	<code>\countdef</code>	332
<code>\@undefined</code>	52	<code>\csname</code>	10, 18, 44, 60, 67, 136, 138, 141, 143, 146, 155, 157, 165, 178, 210, 331, 334, 337, 340, 379, 401
<code>\[</code>	392	<b>E</b>	
<code>\]</code>	122, 388	<code>\empty</code>	13, 14
<code>\{</code>	327, 389		
<code>\}</code>	328, 390		
<code>\]</code>	393		
<code>\_</code>	395		
A			
<code>\advance</code>	368, 376		
<code>\aftergroup</code>	26		



\end .....	402	\ltx@one .....	100
\endcsname .....	10, 18, 44, 60, 67, 136, 138, 141, 143, 146, 155, 157, 165, 177, 178, 210, 331, 334, 337, 340, 379, 401	\ltx@percentchar .....	116
\endinput .....	26	\ltx@pkgextension ....	194, 202, 240
\escapechar .....	128, 133	\ltx@RemovePrefix .....	4, 189, 191
<b>F</b>		\ltx@ReturnAfterElseFi .....	113
\futurelet .....	298, 315	\ltx@ReturnAfterFi .....	3, 112, 113
<b>I</b>		\ltx@secondoftwo .....	109, 160, 168, 181, 196, 216, 231
\if .....	135	\ltx@space .....	3, 115, 230
\ifcase .....	219	\ltx@StripPrefix ..	190, 257, 258, 259
\ifcsname .....	177	\ltx@two .....	101, 297, 308
\iffalse .....	144	\ltx@undefined .....	266, 280
\ifnum ....	220, 222, 224, 251, 367, 375	\ltx@zero .....	2, 99, 296, 302
\iftrue .....	139	\LTxcmds@ifnextchar .....	305, 312
\ifx .	11, 14, 18, 44, 52, 55, 155, 157, 165, 178, 266, 269, 280, 283, 301, 304, 331, 334, 337, 340, 379	\LTxcmds@@ParseVersion ....	244, 246
\immediate .....	20, 46	\LTxcmds@AtEnd .....	80, 81, 324
\input .....	380	\LTxcmds@CharToken .....	295, 301
\iterate .....	348, 350, 352	\LTxcmds@gtemp .....	279, 280, 281, 283, 288, 289, 291
<b>L</b>		\LTxcmds@ifLater .....	206, 218
\lccode .....	117, 122	\LTxcmds@ifnextchar ...	298, 300, 315
\letLTxcmds@gtemp .....	284	\LTxcmds@LetToken .	298, 301, 304, 315
\LoadCommand .....	380, 397	\LTxcmds@newif .....	130, 132
\loop .....	346, 362, 373	\LTxcmds@ParseVersion .....	208, 214, 243, 250
\lowercase .....	118, 123	\LTxcmds@SpaceToken .....	304, 319
\ltx@backslashchar .....	121	<b>N</b>	
\ltx@car .....	3, 110	\next .....	352, 354, 356
\ltx@cclv .....	102	\number .....	207, 213
\ltx@cdr .....	111	<b>P</b>	
\ltx@clsextension ...	4, 193, 199, 237	\PackageInfo .....	23
\ltx@empty ..	3, 114, 267, 270, 281, 284	\pdflastmatch .....	257, 258, 259
\ltx@firstofone .....	2, 107, 173	\pdfmatch .....	251
\ltx@firstoftwo .....	108, 158, 166, 179, 184, 196, 233	\ProvidesPackage .....	15, 61
\ltx@GlobalAppendToMacro .....	5, 265	<b>R</b>	
\ltx@gobble .....	2, 103, 171	\RangeCatcodeInvalid .....	371, 383, 384, 385, 386
\ltx@gobblefour .....	106	\repeat .....	346, 358, 369, 377
\ltx@gobblethree .....	105	\RestoreCatcodes ...	360, 363, 364, 398
\ltx@gobbletwo .....	104	<b>T</b>	
\ltx@ifclasslater .....	4, 236	\Test .....	382, 400
\ltx@ifclassloaded .....	4, 198	\the .....	68, 69, 70, 71, 82, 275, 289, 302, 307, 365
\ltx@iffilelater .....	204, 240	\TMP@EnsureCode ...	79, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98
\ltx@iffileloaded	4, 195, 199, 202, 205	\toks .....	274, 275, 288, 289, 296, 297, 302, 307
\ltx@ifnextchar .....	5, 293	<b>W</b>	
\ltx@ifpackagelater .....	239	\write .....	20, 46
\ltx@ifpackageloaded .....	201	<b>X</b>	
\ltx@ifUndefined ....	3, 163, 187, 242	\x .....	10, 11, 14, 19, 23, 25, 45, 50, 60, 66, 74, 313, 318, 320, 323
\ltx@ifundefined	3, 156, 176, 187, 196		
\ltx@LocalAppendToMacro .....	278		
\ltx@LocalExpandAfter ...	3, 148, 154		
\ltx@newif .....	3, 126		