

The `ltxnew`* package

provides the `\new` `\renew` and `\provide` prefixes for checking definitions.

Florent Chervet <florent.chervet@free.fr>

April 17, 2010

Abstract

`ltxnew` provides `\new` `\renew` and `\provide`: three expandable prefixes for use with `\def`, `\gdef`, `\edef`, `\xdef`, `\countdef`, `\dimendef`, `\skipdef`, `\muskipdef`, `\box`, `\toksdef`, `\marks`, `\count`, `\dimen`, `\skip`, `\muskip`, `\savebox`, `\toks` and the `\glob***` and `\loc***` variants of the `etex` package.

For example:

```
\new\def\macro will do something like: \newcommand\macro{} \def\macro  
\new\let\macro will do something like: \newcommand\macro{} \let\macro
```

...But in fact `\new` does a little more than that... (see [Using `\new`](#)).

You may use `\new` or `\renew` for declaring macros, counters, dimensions, skips, muskips, boxes, tokens and ε -`TEX`'s marks. Even with `\let`, `\new` can be used. Moreover, `\renew` can be used to redefine macros that were previously defined as `\outer`.

`ltxnew` is designed to work with an ε -`TEX` distribution of `LATEX`. It relies on the `LATEX` macro `\@ifdefinable`, on the `etex`¹ package and some macros of `etoolbox`².

Contents

1	Introduction	2
1.1	Motivation	2
1.2	What <code>\new</code> means...	2
1.3	What <code>\renew</code> means	3
1.4	What <code>\provide</code> means	3
1.5	Using <code>\new</code>	3
1.6	Using <code>\renew</code> and <code>\provide</code>	4
2	Implementation	4
2.1	Identification	4
2.2	Requirements	4
2.3	Helper macro	4
2.4	The prefixes scanner	5
2.5	The cancel macros	7
2.6	The defining macros	7
2.7	The prefixes: <code>\new</code> , <code>\renew</code> and <code>\provide</code>	7
3	History	8

* `ltxnew`: [CTAN:macros/latex/contrib/ltxnew](#)

1. `etex`: [CTAN:macros/latex/contrib/etex-pkg](#)

2. `etoolbox`: [CTAN:macros/latex/contrib/etoolbox](#)

This documentation is produced with the `DocStrip` utility.

```
→ To get the documentation, run (thrice): pdflatex ltxnew.dtx  
for the index: makeindex -s gind.ist ltxnew.idx
```

```
→ To get the package, run: etex ltxnew.dtx
```

The `.dtx` file is embedded into this pdf file thank to `embedfile` by H. Oberdiek.

[2010/04/17 v1.2]	8
[2009/10/11 v1.1]	8
[2009/07/22 v1.0]	8
4 References	8
5 Index	8

1 Introduction

1.1 Motivation

L^AT_EX provides `\newcommand` for defining new commands. However, comparing to `\def` the syntax is limited because we cannot use delimited arguments in such a command. The advantage of `\newcommand` (apart the optional argument³) is that the control sequence is first checked for availability (its meaning ought to be `undefined` or `\relax` before the definition).

etoolbox enhance this matter allowing to define `\newrobustcmd` and `\renewrobustcmd`.

Moreover, L^AT_EX does not provide an automatic check of control sequences when defining tokens (`\newtoks`), dimensions (`\newdimen`), skips (`\newskip`), etc. etc.

The only exceptions are:

- `\newlength`
but there is no `\renewlength` command... because the name `\renewlength` sounds bad: it would have meant “*I know the control sequence I wish to define as a length has been defined before, as a macro may be, or a box or a token or whatever, and I wish to redefine this control sequence to be a length (ie a skip)*”. So it doesn’t really make sense...
- `\newcounter`
but `\newcounter{name}` does not define `name` but `\c@name` instead, as a counter.
- `\newsavebox`
- `\newfont`

All those `\new*` stuff define control sequences globally, *excepting* `\newfont`. The reason could to be found in the background⁴.**

But it’s a matter of fact : *fonts* are local to L^AT_EX while *length (ie. skips)* are global...

Thank to the `etex` package that provides a method for the local allocation of new quantity `ltxnew` puts the state of the affairs in a better order. `ltxnew` provides a way to define new control sequences, or redefine them, just by beginning the definition with a (expandable) prefix : `\new` or `\renew`.

1.2 What `\new` means...

Such a short and easy word as `new` ought to be defined !

`\new` means:

- Check if the control sequence to define is available (*ie* means `undefined` or `\relax`)
- If that’s OK: go on (with a side effect if the package tracing is loaded)
- If not : throw an error, and if in `scrollmode` or `nonstopmode` or `batchmode` do not overwrite the last meaning.

That is really what means `\new`. No more, no less.

3. optional arguments are implemented in a much flexible way by `xargs` by Manuel Pégourié-Gonnard.

4. in fact, a new font is defined as a control sequence, just like a macro, whereas `skips`, `dimens`, `tokens` etc. are numbered and then, defining a new one require an allocation.

1.3 What `\renew` means

`\renew` means:

- Check if the control sequence to redefine already has a meaning (different from `undefined` and also from `\relax`)
- If that’s OK : go on (with a side effect if the package tracing is loaded)
- If not : throw an error. But if in `srollmode`, `nonstopmode` or `batchmode` **do define** the control sequence.

1.4 What `\provide` means

`\provide` means:

- Check if the control sequence to define already has a meaning (different from `undefined` and also from `\relax`)
- If that’s OK : go on (with a side effect if the package tracing is loaded)
- If not : silently do nothing.

1.5 Using `\new`

`\new` acts as a (expandable) prefix with the following syntax:

<p style="color: blue; margin: 0;">possibly in a macro</p>	$\left\{ \begin{array}{l} \text{\new} \quad _ \\ (\text{\long_}\text{\global_}\text{\protected_}\text{\outer_}) \\ \text{<DEFINITION WORD>} \\ \text{control sequence} \end{array} \right.$	<p style="margin: 0;">optional (zero or more) required: see below required</p>
--	--	--

`_` denote optional spaces, ignored by the `\new`-prefixes-scanner.

The `<DEFINITION WORD>` may be one of the following:

General:	<code>\let</code>			
Macros:	<code>\def</code>	<code>\gdef</code>	<code>\edef</code>	<code>\xdef</code>
<i>Type</i>	<i>def-word</i>	<i>always global</i>	<i>local (unless \global)</i>	<i>global</i>
Counters:	<code>\countdef</code>	<code>\count</code>	<code>\loccount</code>	<code>\globcount</code>
Dimensions:	<code>\dimendef</code>	<code>\dimen</code>	<code>\locdimen</code>	<code>\globdimen</code>
Skip:	<code>\skipdef</code>	<code>\skip</code> or <code>\length</code>	<code>\locskip</code>	<code>\globskip</code>
Muskip:	<code>\muskipdef</code>	<code>\muskip</code>	<code>\locmuskip</code>	<code>\globmuskip</code>
Box:	<code>\box</code>	<code>\savebox</code>	<code>\locbox</code>	<code>\globbox</code>
Tokens:	<code>\toksdef</code>	<code>\toks</code>	<code>\loctoks</code>	<code>\globtoks</code>
Fonts:	<code>\font</code>			
Marks:	<code>\marks</code>		<code>\locmarks⁵</code>	<code>\globmarks</code>
Write:		<code>\write</code>		
Read:		<code>\read</code>		

Table 1: List of definition-words that may be used with `\new` `\renew` and `\provide`

5. The use of `\locmarks` is left to the appreciation of the user...

2.4 The prefixes scanner

The prefixes scanner is very simple in fact! All the job is based of \futurelet: \futurelet reads the next token but does not remove it from the input string. We then just have to test it with \ifx to conditionally append it into the prefix buffer: \ltxn@prfx. Otherwise, we expand the prefix once and try again. Namely:

```
\futurelet\x\testmacro → if \testmacro “returned false” then:
\expandafter\futurelet\expandafter\x\expandafter\testmacro
```

easy easy easy...

If it happens that the expanded prefix is the same before and after expansion, then it means that was a primitive. The only primitives allowed between \new and \def are:

```
\long          \global          \protected     \outer
\expandafter   \noexpand         and             \relax
```

`\ltxn@prefix` This is the prefix scanner. We open a group at the very beginning for all definitions will be local until the final definition:

```
8 \def\ltxn@prefix{\begingroup
9   \newif\ifglobal
10  \let\ltxn@prfx@empty
11  \let\ltxn@rubbish\relax
12  \futurelet\x\ltxn@@prefix}
```

`\ltxn@@prefix` This is the test macro: it is very long because there are many many \ifx... and as many fees!

```
13 \def\ltxn@@prefix{%
14   \let\ltxn@next@addto\ltxn@next@prefix
15   \ifx\x@\sptoken      \let\next\ltxn@space@prefix%1
16   \else                 \let\next\ltxn@addto@prfx
17     \ifx\x\long        \def\z{\long}%2
18     \else\ifx\x\protected\def\z{\protected}%3
19     \else\ifx\x\global  \let\z@empty\globaltrue%4
20     \else\ifx\x\outer   \def\z{\outer}%5
21     \else
22       \ifx\x\expandafter \def\z{\expandafter}%6
23       \else\ifx\x\noexpand \def\z{\noexpand}%7
24       \else\ifx\x\relax   \def\z{\relax}%8
25       \else
26         \def\ltxn@next@addto{\expandafter\ltxn@def\noexpand}%
27         \ifx\x\let        \def\z{\let}%9
28         \let\ltxn@cancel\ltxn@cancel@let
29         \else            \let\ltxn@cancel\ltxn@cancel@def
30         \ifx\x\def       \edef\z{\ifglobal\global\fi\def}%10
31         \else\ifx\x\edef \edef\z{\ifglobal\global\fi\edef}%11
32         \else\ifx\x\gdef  \def\z{\gdef}%12
33         \else\ifx\x\xdef  \def\z{\xdef}%13
34         \else            \let\ltxn@cancel\ltxn@cancel@new
35         \ifx\x\count     \def\z{\newcount}%14
36         \else\ifx\x\countdef%15
37           \ifglobal\def\z{\globcount}\else\def\z{\loccount}\fi
38         \else\ifx\x\loccount%16
39           \ifglobal\def\z{\globcount}\else\def\z{\loccount}\fi
40         \else\ifx\x\globcount \def\z{\globcount}%17
41         \else\ifx\x\dimen    \def\z{\newdimen}%18
42         \else\ifx\x\dimendef%19
43           \ifglobal\def\z{\globdimen}\else\def\z{\locdimen}\fi
44         \else\ifx\x\locdimen%20
45           \ifglobal\def\z{\globdimen}\else\def\z{\locdimen}\fi
46         \else\ifx\x\globdimen \def\z{\globdimen}%21
47         \else\ifx\x\skip     \def\z{\newskip}%22
48         \else\ifx\x\skipdef%23
```

```

49         \ifglobal\def\z{\globskip}\else\def\z{\locskip}\fi
50     \else\ifx\x\locskip%%24
51         \ifglobal\def\z{\globskip}\else\def\z{\locskip}\fi
52     \else\ifx\x\globskip    \def\z{\globskip}%%25
53     \else\ifx\x\muskip      \def\z{\newmuskip}%%26
54     \else\ifx\x\muskipdef%%27
55         \ifglobal\def\z{\globmuskip}\else\def\z{\locmuskip}\fi
56     \else\ifx\x\locmuskip%%28
57         \ifglobal\def\z{\globmuskip}\else\def\z{\locmuskip}\fi
58     \else\ifx\x\globmuskip  \def\z{\globmuskip}%%29
59     \else\ifx\x\savebox     \def\z{\newsavebox}%%30
60     \else\ifx\x\box%%31
61         \ifglobal\def\z{\globbox}\else\def\z{\locbox}\fi
62     \else\ifx\x\locbox%%
63 32
64         \ifglobal\def\z{\globbox}\else\def\z{\locbox}\fi
65     \else\ifx\x\globbox     \def\z{\globbox}%%33
66     \else\ifx\x\toksdef%%34
67         \ifglobal\def\z{\globtoks}\else\def\z{\loctoks}\fi
68     \else\ifx\x\toks       \def\z{\newtoks}%%35
69     \else\ifx\x\loctoks%%36
70         \ifglobal\def\z{\globtoks}\else\def\z{\loctoks}\fi
71     \else\ifx\x\globtoks   \def\z{\globtoks}%%37
72     \else\ifx\x\locmarks%%38
73         \ifglobal\def\z{\globmarks}\else\def\z{\locmarks}\fi
74     \else\ifx\x\marks      \def\z{\newmarks}%%39    %\newmarks=\globmarks
75     \else\ifx\x\globmarks  \def\z{\globmarks}%%40
76     \else\ifx\x\font       \def\z{\font}%%41
77     \else\ifx\x\write      \def\z{\newwrite}%%42
78     \else\ifx\x\read       \def\z{\newread}%%43
79     \else\ifx\x\protect    \ltxn@error@prefix%%44
80     \else
81         \let\ltxn@next@addto\ltxn@next@prefix
82         \ifx\y\x\ltxn@error@prefix
83         \else\let\y\x
84         \fi
85         \let\next\ltxn@expand@prefix
86         \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
87         \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
88         \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
89     \fi\fi\fi\fi
90     \fi
91     \fi\fi\fi
92     \fi\fi\fi\fi% so many fees...
93     \fi\next}

94 \def\ltxn@next@prefix{\futurelet\x\ltxn@@@prefix}
95 \def\ltxn@expand@prefix{%
96     \expandafter\futurelet\expandafter\x\expandafter\ltxn@@@prefix}
97 \def\ltxn@addto@prfx#1{\let\y\@undefined
98     \expandafter\expandafter\expandafter\def
99         \expandafter\expandafter\expandafter\ltxn@prfx
100         \expandafter\expandafter\expandafter{\expandafter\ltxn@prfx\z}%
101 %   \edef\ltxn@prfx{\ltxn@expandonce\ltxn@prfx\ltxn@expandonce\z}%
102     \ltxn@next@addto}
103 \expandafter\def\expandafter\ltxn@space@prefix\space{\ltxn@next@prefix}
104 \def\ltxn@error@prefix{\@latex@error{A \string\def\space
105     (or \string\countdef\space or\string\toksdef\space etc.)\MessageBreak
106     was expected after \string\new\MessageBreak
107     I found a \meaning\x!\MessageBreak
108     see ltxnew documentation for more information}}\@ehd}

```

2.5 The cancel macros

`\ltx@cancel` These are the macros used in case we have to cancel definition (nonstopmode)

```

109 \def\ltx@cancel@let{\afterassignment\endgroup\let\ltx@rubbish}
110 \def\ltx@cancel@def{\afterassignment\endgroup\def\ltx@rubbish}
111 \def\ltx@cancel@new{\endgroup}

```

2.6 The defining macros

`\ltx@new` `\ltx@new` defines the new control sequence, or cancels definition depending on the result of `\ifdefinable`. `\ltx@new` does all the jobs: it is called by both `\ltx@renew` and `\ltx@provide`:

```

112 \def\ltx@new#1{%
113   \let\next\ltx@cancel
114   \ifdefined#1\unless\ifx#1\relax\def#1{new:error}\fi\fi
115   \expandafter\@ifdefinable\noexpand#1{%
116     \expandafter\let\noexpand#1=\relax
117     \edef\next{\endgroup\ltx@expandonce\ltx@prfx#1}%
118   \next}
119

```

`\ltx@renew` `\ltx@renew` throws an error if the control sequence is undefined or if its meaning is `\relax`. In case of nonstopmode the control sequence is redefined, however. To handle the case where the control sequence was an outer macro, we “stringify” its name first, in order not to give the control sequence itself as an argument for the error message.

```

120 \def\ltx@renew#1{%
121   \edef\ltx@name{\string#1}%
122   \ifdefined#1\ifx#1\relax\ltx@error{renew: \ltx@name undefined}\fi
123   \else      \ltx@error{renew: \ltx@name undefined}%
124   \fi
125   \let#1=\relax
126   \def\next{\ltx@new#1}%
127   \next}

```

`\ltx@provide` `\ltx@provide` never throws an error, but define the control sequence only if it is undefined or `\relax` (*ie* if it is definable):

To handle the case where the control sequence was an outer macro, we “stringify” its name first, in order not to put the control sequence itself in the definition of `\next`. It’s admittedly tricky here, because if the control sequence is already defined, `\provide` will cancel out the new definition, however, as a borderline effect, `\ltx@new` should have been called with this very control sequence as an argument, if it had been definable. Even if this `\iffalse`-call (not expanded) is prepared into `\ifx...\fi` conditional, the `\outer` control sequence is there, and T_EX (not L^AT_EX) will throw an error: Forbidden control sequence found....

```

128 \def\ltx@provide#1{%
129   \let\next\ltx@cancel
130   \edef\ltx@name{\string#1}%
131   \ifdefined#1\ifx#1\relax \ltx@provide@new\fi
132   \else \ltx@provide@new
133   \fi
134   \next}
135 \def\ltx@provide@new{%
136   \edef\next{\noexpand\ltx@new\cename\expandafter\@gobble\ltx@name\endcename}}

```

2.7 The prefixes: `\new`, `\renew` and `\provide`

`\new` `\new`: the entry point: just let the definition macro to be `\ltx@new` and start scanning prefixes.

```

137 \protected\def\new{\let\ltx@def\ltx@new\ltx@prefix}

```


ltxnew – provides the `\new` `\renew` and `\provide` prefixes for checking definitions.

`\renew` `\renew`: the entry point: just let the definition macro to be `\ltxn@renew` and start scanning prefixes.

```
138 \protected\def\renew{\let\ltxn@def\ltxn@renew\ltxn@prefix}
```

`\provide` `\provide`: the entry point: just let the definition macro to be `\ltxn@provide` and start scanning prefixes.

```
139 \protected\def\provide{\let\ltxn@def\ltxn@provide\ltxn@prefix}
```

`\ltxn@error` In case of redefinition, throws an `\ehc`-type error:

```
140 \def\ltxn@error#1{\@latex@error{#1}\@ehc}
```

```
141 \</package>
```

3 History

[2010/04/17 v1.2]

- `\provide` and `\renew` added an undesirable blank space.

[2009/10/11 v1.1]

- Correction of `.sty` header.

[2009/07/22 v1.0]

- First version.

4 References

[1] David Carlisle and Peter Breitenlohner *The etex package*; 1998/03/26 v2.0; CTAN:macros/latex/contrib/etex-pkg/.

[2] Philipp Lehman *The etoolbox package*; 2008/06/28 v1.7; CTAN:macros/latex/contrib/etoolbox/.

5 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			
<code>\@ehc</code>	140	<code>\dimendef</code>	42
<code>\@ehd</code>	108		F
<code>\@ifdefinable</code>	115	<code>\font</code>	76
<code>\@sptoken</code>	15	<code>\futurelet</code>	12, 94, 96
<code>\@undefined</code>	97		G
	A	<code>\globaltrue</code>	19
<code>\afterassignment</code>	109, 110	<code>\globbox</code>	61, 64, 65
	C	<code>\globcount</code>	37, 39, 40
<code>\count</code>	35	<code>\globdimen</code>	43, 45, 46
<code>\countdef</code>	36, 105	<code>\globmarks</code>	73, 74, 75
	D	<code>\globmuskip</code>	55, 57, 58
<code>\dimen</code>	41	<code>\globskip</code>	49, 51, 52
		<code>\globtoks</code>	67, 70, 71

I		\newmarks	74
\ifglobal	9, 30, 31, 37, 39, 43, 45, 49, 51, 55, 57, 61, 64, 67, 70, 73	\newmuskip	53
L		\newread	78
\locbox	61, 62, 64	\newsavebox	59
\loccount	37, 38, 39	\newskip	47
\locdimen	43, 44, 45	\newtoks	68
\locmarks	72, 73	\newwrite	77
\locmuskip	55, 56, 57	P	
\locskip	49, 50, 51	\protected	18, 137, 138, 139
\loctoks	67, 69, 70	\provide	<u>139</u>
\ltx@cancel	<u>109</u>	R	
\ltxn@prefix	12, <u>13</u> , 94, <u>96</u>	\read	78
\ltxn@addto@prfx	16, 97	\renew	<u>138</u>
\ltxn@cancel	28, 29, 34, 113, 129	S	
\ltxn@cancel@def	29, 110	\savebox	59
\ltxn@cancel@let	28, 109	\skip	47
\ltxn@cancel@new	34, 111	\skipdef	48
\ltxn@def	26, 137, 138, 139	T	
\ltxn@error	122, 123, <u>140</u>	\toks	68
\ltxn@error@prefix	79, 82, <u>104</u>	\toksdef	66, 105
\ltxn@expand@prefix	85, 95	U	
\ltxn@expandonce	7, 101, 117	\unexpanded	7
\ltxn@name	121, 122, 123, 130, 136	\unless	114
\ltxn@new	<u>112</u> , 126, 136, 137	W	
\ltxn@next@addto	14, 26, 81, 102	\write	77
\ltxn@next@prefix	14, 81, 94, 103	X	
\ltxn@prefix	8, 137, 138, 139	\x	12, 15, 17, 18, 19, 20, 22, 23, 24, 27, 30, 31, 32, 33, 35, 36, 38, 40, 41, 42, 44, 46, 47, 48, 50, 52, 53, 54, 56, 58, 59, 60, 62, 65, 66, 68, 69, 71, 72, 74, 75, 76, 77, 78, 79, 82, 83, 94, 96, 107
\ltxn@prfx	10, 99, 100, 101, 117	Y	
\ltxn@provide	<u>128</u> , 139	\y	82, 83, 97
\ltxn@provide@new	131, 132, 135	Z	
\ltxn@renew	<u>120</u> , 138	\z	17, 18, 19, 20, 22, 23, 24, 27, 30, 31, 32, 33, 35, 37, 39, 40, 41, 43, 45, 46, 47, 49, 51, 52, 53, 55, 57, 58, 59, 61, 64, 65, 67, 68, 70, 71, 73, 74, 75, 76, 77, 78, 100, 101
\ltxn@rubbish	11, 109, 110		
\ltxn@space@prefix	15, 103		
M			
\marks	74		
\meaning	107		
\muskip	53		
\muskipdef	54		
N			
\new	106, <u>137</u>		
\newcount	35		
\newdimen	41		