

The l3flag package: expandable flags*

The L^AT_EX3 Project[†]

Released 2012/02/08

Flags are the only data-type on which T_EX can perform assignments in expansion-only contexts. This module is meant mostly for kernel use: in almost all cases, booleans or integers should be preferred to flags, because they are faster.

A flag can hold any non-negative value, which we call its *height*. In expansion-only contexts, a flag can only be “raised”: this normally increases the *height* by 1, but can be configured by defining specific traps. The *height* can also be queried expandably. However, decreasing it, or setting it to zero requires non-expandable assignments.

Flag variables are always local. They are referenced by a *name* of the form *package_flag name*, for instance, `str_missing`.

1 Setting up flags

<code>\flag_new:n</code>	<code>\flag_new:n {<flag name>}</code>
--------------------------	--

Creates a new *flag* with a name given by *flag name*, or raises an error if the name is already taken. The *flag name* must consist of character tokens only. The declaration is global, but flags are always local variables. The *flag* will initially have zero height.

<code>\flag_clear:n</code>	<code>\flag_clear:n {<flag name>}</code>
----------------------------	--

The *flag*’s height is set to zero. The assignment is local.

<code>\flag_clear_new:n</code>	<code>\flag_clear_new:n {<flag name>}</code>
--------------------------------	--

Ensures that the *flag* exists globally by applying `\flag_new:n` if necessary, then applies `\flag_zero:n`, setting the height to zero locally.

<code>\flag_set_trap:nn</code>	<code>\flag_set_trap:nn {<flag name>} {<inline function>}</code>
--------------------------------	--

Changes the action that is taken when the *flag* is raised using `\flag_raise:n`. Instead of the default action which is to increase the *flag*’s height by 1, the *inline function* will be called, receiving the current flag’s height as `#1`. The *inline function* should expand to nothing; *e.g.*, it could call `\msg_expandable_error:n`. This function is very experimental.

*This file describes v3342, last revised 2012/02/08.

[†]E-mail: latex-team@latex-project.org

2 Expandable flag commands

<hr/> <code>\flag_if_exist_p:n</code> ★	<code>\flag_if_exist:n {⟨flag name⟩}</code>
<code>\flag_if_exist:nTF</code> ★	This function returns <code>true</code> if the <code>⟨flag name⟩</code> references a flag that has been defined previously, and <code>false</code> otherwise.
<hr/> <code>\flag_if_raised_p:n</code> ★	<code>\flag_if_raised:n {⟨flag name⟩}</code>
<code>\flag_if_raised:nTF</code> ★	This function returns <code>true</code> if the <code>⟨flag⟩</code> has non-zero height, and <code>false</code> if the <code>⟨flag⟩</code> has zero height.
<hr/> <code>\flag_height:n</code> ★	<code>\flag_height:n {⟨flag name⟩}</code>
	Expands to the height of the <code>⟨flag⟩</code> as an integer denotation.
<hr/> <code>\flag_raise:n</code> ★	<code>\flag_raise:n {⟨flag name⟩}</code>
	The <code>⟨flag⟩</code> 's trap is performed, taking the current height as its argument. The default behaviour is to increase the <code>⟨flag⟩</code> 's height by 1 locally. This function is expandable, as long as the trap is expandable (the default trap is expandable, despite being an assignment).

3 l3flag implementation

```

1 ⟨*initex | package⟩
2 \ProvidesExplPackage
3   {⟨ExplFileName⟩}{⟨ExplFileDate⟩}{⟨ExplFileVersion⟩}{⟨ExplFileDescription⟩}

```

3.1 Non-expandable flag commands

<code>\flag_new:n</code>	For each flag, we define a “trap” function, which by default simply increases the flag by 1.
<code>\cs_new_protected:Npn \flag_new:n #1</code>	
<code>{</code>	
<code>\cs_new:cpn { flag_trap_#1:w } ##1 ;</code>	
<code>{ \exp_after:wN \use_none:n \cs:w l_#1_##1_flag \cs_end: }</code>	
<code>}</code>	
	<i>(End definition for <code>\flag_new:n</code>. This function is documented on page 1.)</i>
<code>\flag_clear:n</code>	Undefine control sequences, starting from the <code>_0</code> flag, upwards, until reaching an undefined control sequence.
<code>\flag_clear_aux:ww</code>	
<code>\cs_new_protected:Npn \flag_clear:n #1</code>	
<code>{ \flag_clear_aux:ww 0 ; #1 \q_stop }</code>	
<code>\cs_new_protected:Npn \flag_clear_aux:ww #1 ; #2 \q_stop</code>	
<code>{</code>	
<code>\if_cs_exist:w l_#2_#1_flag \cs_end:</code>	
<code>\else:</code>	
<code>\exp_after:wN \use_none_delimit_by_q_stop:w</code>	

```

16 \fi:
17 \cs_set_eq:cN { l_#2_#1_flag } \c_undefined:D
18 \exp_after:wN \flag_clear_aux:ww
19 \int_use:N \int_eval:w \c_one + #1 ;
20 #2 \q_stop
21 }

```

(End definition for \flag_clear:n. This function is documented on page ??.)

\flag_clear_new:n A flag exist if \flag_trap_⟨flag name⟩:n exists.

```

22 \cs_new_protected:Npn \flag_clear_new:n #1
23 { \flag_if_exist:nTF {#1} { \flag_clear:n } { \flag_new:n } {#1} }

```

(End definition for \flag_clear_new:n. This function is documented on page 1.)

\flag_set_trap:nn Should that l3flag function check whether the flag exists?

```

24 \cs_new_protected:Npn \flag_set_trap:nn #1#2
25 { \cs_set:cpn { flag_trap_#1:w } ##1 ; {#2} }

```

(End definition for \flag_set_trap:nn. This function is documented on page 1.)

3.2 Expandable flag commands

\flag_if_exist:n A ⟨flag⟩ is defined if the corresponding “trap” is defined.

```

26 \prg_new_conditional:Npnn \flag_if_exist:n #1 { p , T , F , TF }
27 {
28   \cs_if_exist:cTF { flag_trap_#1:w }
29   { \prg_return_true: } { \prg_return_false: }
30 }

```

(End definition for \flag_if_exist:n. This function is documented on page 2.)

\flag_if_raised:n Test if the flag is non-zero, by checking the _0 control sequence.

```

31 \prg_new_conditional:Npnn \flag_if_raised:n #1 { p , T , F , TF }
32 {
33   \if_cs_exist:w l_#1_0_flag \cs_end:
34   \prg_return_true:
35   \else:
36   \prg_return_false:
37   \fi:
38 }

```

(End definition for \flag_if_raised:n. This function is documented on page 2.)

\flag_height:n Extract the value of the flag by going through all of the _⟨integer⟩ control sequences starting from 0.

```

\flag_height_loop:ww
\flag_height_end:ww
39 \cs_new:Npn \flag_height:n #1 { \flag_height_loop:ww 0; #1 \q_stop }
40 \cs_new:Npn \flag_height_loop:ww #1 ; #2 \q_stop
41 {
42   \if_cs_exist:w l_#2_#1_flag \cs_end:
43   \exp_after:wN \flag_height_loop:ww \int_use:N \int_eval:w \c_one +
44   \else:
45   \exp_after:wN \flag_height_end:ww

```

```

46     \fi:
47     #1 ; #2 \q_stop
48   }
49 \cs_new:Npn \flag_height_end:ww #1 ; #2 \q_stop { #1 }
      (End definition for \flag_height:n. This function is documented on page ??.)

\flag_raise:n Simply apply the trap to the height, after expanding the latter.

50 \cs_new:Npn \flag_raise:n #1
51   {
52     \cs:w flag_trap_#1:w \exp_after:wN \cs_end:
53     \int_value:w \flag_height:n {#1} ;
54   }
      (End definition for \flag_raise:n. This function is documented on page 2.)

55 </initex | package)

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

C		
\c_one	19, 43	\flag_if_exist:n 26, 26
\c_undefined:D	17	\flag_if_exist:nTF 2, 23
\cs:w	7, 52	\flag_if_raised:n 31, 31
\cs_end:	7, 13, 33, 42, 52	\flag_if_raised:nTF 2
\cs_if_exist:cTF	28	\flag_new:n 1, 4, 4, 23
\cs_new:cpn	6	\flag_raise:n 2, 50, 50
\cs_new:Npn	39, 40, 49, 50	\flag_set_trap:nn 1, 24, 24
\cs_new_protected:Npn ...	4, 9, 11, 22, 24	
\cs_set:cpn	25	
\cs_set_eq:cN	17	
E		
\else:	14, 35, 44	
\exp_after:wN	7, 15, 18, 43, 45, 52	
\ExplFileDate	3	
\ExplFileDescription	3	
\ExplFileName	3	
\ExplFileVersion	3	
F		
\fi:	16, 37, 46	
\flag_clear:n	1, 9, 9, 23	
\flag_clear_aux:ww	9, 10, 11, 18	
\flag_clear_new:n	1, 22, 22	
\flag_height:n	2, 39, 39, 53	
\flag_height_end:ww	39, 45, 49	
\flag_height_loop:ww	39, 39, 40, 43	
I		
\if_cs_exist:w	13, 33, 42	
\int_eval:w	19, 43	
\int_use:N	19, 43	
\int_value:w	53	
L		
\l_doc_pTF_name_tl	2	
P		
\prg_new_conditional:Npnn	26, 31	
\prg_return_false:	29, 36	
\prg_return_true:	29, 34	
\ProvidesExplPackage	2	
Q		
\q_stop	10, 11, 20, 39, 40, 47, 49	
U		
\use_none:n	7	
\use_none_delimit_by_q_stop:w	15	