

# The **l3flag** package: expandable flags\*

The L<sup>A</sup>T<sub>E</sub>X3 Project<sup>†</sup>

Released 2011/12/08

Flags are the only data-type on which T<sub>E</sub>X 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 {&lt;flag name&gt;}</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.

function

---

---

<code>\flag_clear:n</code>	<code>\flag_clear:n {&lt;flag name&gt;}</code>
----------------------------	--

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

function

---

---

<code>\flag_clear_new:n</code>	<code>\flag_clear_new:n {&lt;flag name&gt;}</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.

function

---

\*This file describes v3039, last revised 2011/12/08.

<sup>†</sup>E-mail: [latex-team@latex-project.org](mailto:latex-team@latex-project.org)

---

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

---

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

function

## 2 Expandable flag commands

---

<code>\flag_if_exist_p:n</code> ★	<code>\flag_if_exist:n {⟨flag name⟩}</code>
<code>\flag_if_exist:nTF</code> ★	

---

This function returns `true` if the  $\langle flag name \rangle$  references a flag that has been defined previously, and `false` otherwise.

XP,pTF

---

<code>\flag_if_raised_p:n</code> ★	<code>\flag_if_raised:n {⟨flag name⟩}</code>
<code>\flag_if_raised:nTF</code> ★	

---

This function returns `true` if the  $\langle flag \rangle$  has non-zero height, and `false` if the  $\langle flag \rangle$  has zero height.

XP,pTF

---

<code>\flag_height:n</code> ★	<code>\flag_height:n {⟨flag name⟩}</code>
-------------------------------	---

---

Expands to the height of the  $\langle flag \rangle$  as an integer denotation.

XP

---

<code>\flag_raise:n</code> ★	<code>\flag_raise:n {⟨flag name⟩}</code>
------------------------------	--

---

The  $\langle flag \rangle$ 's trap is performed, taking the current height as its argument. The default behaviour is to increase the  $\langle flag \rangle$ '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).

XP

## 3 l3flag implementation

```

1 <*initex | package>
2 <@@=flag>
3 \ProvidesExplPackage
4   {\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}

```

### 3.1 Non-expandable flag commands

`\flag_new:n` For each flag, we define a “trap” function, which by default simply increases the flag by 1.

```

5 \cs_new_protected:Npn \flag_new:n #1

```

```

6 {
7   \cs_new:cpn { __flag_trap_#1:w } ##1 ;
8   { \exp_after:wN \use_none:n \cs:w __flag_#1_##1: \cs_end: }
9 }

```

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

**\flag\_clear:n** **\\_\_flag\_clear:ww** Undefine control sequences, starting from the `_0` flag, upwards, until reaching an undefined control sequence.

```

10 \cs_new_protected:Npn \flag_clear:n #1
11 { \__flag_clear:ww 0 ; #1 \q_stop }
12 \cs_new_protected:Npn \__flag_clear:ww #1 ; #2 \q_stop
13 {
14   \if_cs_exist:w __flag_#2_#1: \cs_end:
15   \else:
16     \exp_after:wN \use_none_delimit_by_q_stop:w
17     \fi:
18     \cs_set_eq:cN { __flag_#2_#1: } \c_undefined:D
19     \exp_after:wN \__flag_clear:ww
20     \int_use:N \__int_eval:w \c_one + #1 ;
21     #2 \q_stop
22 }

```

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

**\flag\_clear\_new:n** As for other datatypes, clear the  $\langle flag \rangle$  or create a new one, as appropriate.

```

23 \cs_new_protected:Npn \flag_clear_new:n #1
24 { \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** Redefine the trap.

```

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

```

(End definition for \flag\_set\_trap:nn This function is documented on page 2.)

## 3.2 Expandable flag commands

**\flag\_if\_exist:p:n** A flag exist if the corresponding trap `\__flag_trap_⟨flag name⟩:n` is defined.

**\flag\_if\_exist:nTF**

```

27 \prg_new_conditional:Npnn \flag_if_exist:n #1 { p , T , F , TF }
28 {
29   \cs_if_exist:cTF { __flag_trap_#1:w }
30   { \prg_return_true: } { \prg_return_false: }
31 }

```

(End definition for \flag\_if\_exist:n These functions are documented on page 2.)

**\flag\_if\_raised:p:n** Test if the flag is non-zero, by checking the `_0` control sequence.

**\flag\_if\_raised:nTF**

```

32 \prg_new_conditional:Npnn \flag_if_raised:n #1 { p , T , F , TF }
33 {
34   \if_cs_exist:w __flag_#1_0: \cs_end:
35   \prg_return_true:

```

```

36     \else:
37         \prg_return_false:
38     \fi:
39 }

```

(End definition for `\flag_if_raised:n` These functions are 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
40 \cs_new:Npn \flag_height:n #1 { \__flag_height_loop:ww 0; #1 \q_stop }
41 \cs_new:Npn \__flag_height_loop:ww #1 ; #2 \q_stop
42 {
43     \if_cs_exist:w __flag_#2_#1: \cs_end:
44     \exp_after:wN \__flag_height_loop:ww \int_use:N \__int_eval:w \c_one +
45     \else:
46     \exp_after:wN \__flag_height_end:ww
47     \fi:
48     #1 ; #2 \q_stop
49 }
50 \cs_new:Npn \__flag_height_end:ww #1 ; #2 \q_stop { #1 }

```

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

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

```

51 \cs_new:Npn \flag_raise:n #1
52 {
53     \cs:w __flag_trap_#1:w \exp_after:wN \cs_end:
54     \__int_value:w \flag_height:n {#1} ;
55 }

```

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

56 `\</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.

Symbols			
<code>\__flag_clear:ww</code>	10, 11, 12, 19	<code>\cs_end:</code>	8, 14, 34, 43, 53
<code>\__flag_height_end:ww</code>	40, 46, 50	<code>\cs_if_exist:cTF</code>	29
<code>\__flag_height_loop:ww</code>	40, 40, 41, 44	<code>\cs_new:cpn</code>	7
<code>\__int_eval:w</code>	20, 44	<code>\cs_new:Npn</code>	40, 41, 50, 51
<code>\__int_value:w</code>	54	<code>\cs_new_protected:Npn</code>	5, 10, 12, 23, 25
		<code>\cs_set:cpn</code>	26
		<code>\cs_set_eq:cN</code>	18
C			
<code>\c_one</code>	20, 44	E	
<code>\c_undefined:D</code>	18	<code>\else:</code>	15, 36, 45
<code>\cs:w</code>	8, 53	<code>\exp_after:wN</code>	8, 16, 19, 44, 46, 53

\ExplFileDate .....	4	\flag_set_trap:nn .....	2, 25, 25
\ExplFileDescription .....	4		
\ExplFileName .....	4	I	
\ExplFileVersion .....	4	\if_cs_exist:w .....	14, 34, 43
		\int_use:N .....	20, 44
F			
\fi: .....	17, 38, 47	P	
\flag_clear:n .....	1, 10, 10, 24	\prg_new_conditional:Npnn .....	27, 32
\flag_clear_new:n .....	1, 23, 23	\prg_return_false: .....	30, 37
\flag_height:n .....	2, 40, 40, 54	\prg_return_true: .....	30, 35
\flag_if_exist:n .....	27	\ProvidesExplPackage .....	3
\flag_if_exist:nTF .....	2, 24, 27		
\flag_if_exist_p:n .....	2, 27	Q	
\flag_if_raised:n .....	32	\q_stop .....	11, 12, 21, 40, 41, 48, 50
\flag_if_raised:nTF .....	2, 32		
\flag_if_raised_p:n .....	2, 32	U	
\flag_new:n .....	1, 5, 5, 24	\use_none:n .....	8
\flag_raise:n .....	2, 51, 51	\use_none_delimit_by_q_stop:w .....	16