The l3flag package: expandable flags[*]

The LaTeX3 Project[†]

Released 2011/12/08

Flags are the only data-type on which TeX 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

`\flag_new:n`

`\flag_new:n {⟨flag name⟩}`

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.

`\flag_clear:n`

`\flag_clear:n {⟨flag name⟩}`

The ⟨*flag*⟩'s height is set to zero. The assignment is local.

`\flag_clear_new:n`

`\flag_clear_new:n {⟨flag name⟩}`

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

`\flag_set_trap:nn`

`\flag_set_trap:nn {⟨flag name⟩} {⟨inline function⟩}`

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 v3039, last revised 2011/12/08.

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

2 Expandable flag commands

`\flag_if_exist_p:n` ⋆
`\flag_if_exist:nTF` ⋆

`\flag_if_exist:n {⟨flag name⟩}`

This function returns `true` if the ⟨*flag name*⟩ references a flag that has been defined previously, and `false` otherwise.

`\flag_if_raised_p:n` ⋆
`\flag_if_raised:nTF` ⋆

`\flag_if_raised:n {⟨flag name⟩}`

This function returns `true` if the ⟨*flag*⟩ has non-zero height, and `false` if the ⟨*flag*⟩ has zero height.

`\flag_height:n` ⋆

`\flag_height:n {⟨flag name⟩}`

Expands to the height of the ⟨*flag*⟩ as an integer denotation.

`\flag_raise:n` ⋆

`\flag_raise:n {⟨flag name⟩}`

The ⟨*flag*⟩'s trap is performed, taking the current height as its argument. The default behaviour is to increase the ⟨*flag*⟩'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

```
⟨*initex | package⟩
⟨@@=flag⟩
\ProvidesExplPackage
  {\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.

```
\cs_new_protected:Npn \flag_new:n #1
  {
    \cs_new:cpn { __flag_trap_#1:w } ##1 ;
      { \exp_after:wN \use_none:n \cs:w __flag_#1_##1: \cs_end: }
  }
```

(*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.

```
\cs_new_protected:Npn \flag_clear:n #1
  { \__flag_clear:ww 0 ; #1 \q_stop }
\cs_new_protected:Npn \__flag_clear:ww #1 ; #2 \q_stop
  {
    \if_cs_exist:w __flag_#2_#1: \cs_end:
```

```
    \else:
      \exp_after:wN \use_none_delimit_by_q_stop:w
    \fi:
    \cs_set_eq:cN { __flag_#2_#1: } \c_undefined:D
    \exp_after:wN \__flag_clear:ww
    \int_use:N \__int_eval:w \c_one + #1 ;
    #2 \q_stop
  }
```

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

`\flag_clear_new:n`

As for other datatypes, clear the ⟨*flag*⟩ or create a new one, as appropriate.

```
\cs_new_protected:Npn \flag_clear_new:n #1
  { \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.

```
\cs_new_protected:Npn \flag_set_trap:nn #1#2
  { \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_p:n`
`\flag_if_exist:nTF`

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

```
\prg_new_conditional:Npnn \flag_if_exist:n #1 { p , T , F , TF }
  {
    \cs_if_exist:cTF { __flag_trap_#1:w }
      { \prg_return_true: } { \prg_return_false: }
  }
```

(*End definition for* `\flag_if_exist:n` *These functions are documented on page 2.*)

`\flag_if_raised_p:n`
`\flag_if_raised:nTF`

Test if the flag is non-zero, by checking the `_0` control sequence.

```
\prg_new_conditional:Npnn \flag_if_raised:n #1 { p , T , F , TF }
  {
    \if_cs_exist:w __flag_#1_0: \cs_end:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(*End definition for* `\flag_if_raised:n` *These functions are documented on page 2.*)

`\flag_height:n`
`\__flag_height_loop:ww`
`\__flag_height_end:ww`

Extract the value of the flag by going through all of the `_`⟨*integer*⟩ control sequences starting from 0.

```
\cs_new:Npn \flag_height:n #1 { \__flag_height_loop:ww 0; #1 \q_stop }
\cs_new:Npn \__flag_height_loop:ww #1 ; #2 \q_stop
  {
    \if_cs_exist:w __flag_#2_#1: \cs_end:
      \exp_after:wN \__flag_height_loop:ww \int_use:N \__int_eval:w \c_one +
    \else:
      \exp_after:wN \__flag_height_end:ww
    \fi:
    #1 ; #2 \q_stop
  }
\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.

```
\cs_new:Npn \flag_raise:n #1
  {
    \cs:w __flag_trap_#1:w \exp_after:wN \cs_end:
    \__int_value:w \flag_height:n {#1} ;
  }
```

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

```
⟨/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.