The l3flag package: expandable flags[*]

The LaTeX3 Project[†]

Released 2012/01/19

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

A flag can be in two states: "lowered" or "raised". The status of a flag can be tested expandably (just like booleans). A flag can also be *raised* expandably. However, a flag cannot be lowered expandably.

Flag variables are always local.

Testing for the existence of flags with `\cs_if_exist:NTF` will always result in a negative answer: from the point of view of TeX, flags are either undefined control sequences, or let to `\scan_stop:`. This has a second consequence: flags don't need to be declared with `\flag_new:N`, although this has error-checking benefits. The ability of using flags without declaring them is used in l3rand.

`\flag_new:N`
`\flag_new:c`

`\flag_new:N ⟨flag⟩`

Creates a new ⟨*flag*⟩ or raises an error if the name is already taken. The declaration is global, but ⟨*flags*⟩ are always local variables. The ⟨*flag*⟩ will initially be lowered.

`\flag_test_p:N` ⋆
`\flag_test_p:c` ⋆
`\flag_test:NTF` ⋆
`\flag_test:cTF` ⋆

`\flag_test:N ⟨flag⟩`

This function returns `true` if the ⟨*flag*⟩ is raised, and `false` if the ⟨*flag*⟩ is lowered.

`\flag_raise:N` ⋆

`\flag_raise:N ⟨flag⟩`

The ⟨*flag*⟩ is raised. The assignment is local. This function is expandable, despite being an assignment.

`\flag_lower:N`

`\flag_lower:N ⟨flag⟩`

The ⟨*flag*⟩ is lowered. The assignment is local.

[*]This file describes v3209, last revised 2012/01/19.
[†]E-mail: latex-team@latex-project.org

1 l3str implementation

```
⟨*initex | package⟩
\ProvidesExplPackage
  {\ExplFileName}{\ExplFileDate}{\ExplFileVersion}{\ExplFileDescription}
```

\g_flag_list_tl

We keep track of the list of all defined flags in a token list. This is only used by \flag_-new:N to check that the flag has not yet been defined.

```
\tl_new:N \g_flag_list_tl
```

(*End definition for* \g_flag_list_tl. *This function is documented on page* **??**.)

\flag_new:N
\flag_new:c

If the control sequence is already defined, then we raise an error. Otherwise we check if it is already a flag, raising an appropriate error, finally lower the flag globally (this is the only global assignment to a flag... perhaps an error?). The c variant can be created without worrying for this function.

```
\cs_new_protected:Npn \flag_new:N #1
  {
    \cs_new_eq:NN #1 \c_undefined:D
    \tl_if_in:NnTF \g_flag_list_tl {#1}
      {
        \msg_kernel_error:nnx
          { flag } { already-defined }
          { \token_to_str:N #1 }
      }
      { \tl_gput_right:Nn \g_flag_list_tl {#1} }
  }
\cs_generate_variant:Nn \flag_new:N { c }
```

(*End definition for* \flag_new:N *and* \flag_new:c. *These functions are documented on page* **??**.)

\flag_test:N
\flag_test:c

Test if the control sequence is undefined or not.

```
\prg_new_conditional:Npnn \flag_test:N #1 { p , T , F , TF }
  {
    \if_cs_exist:N #1
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
\prg_new_conditional:Npnn \flag_test:c #1 { p , T , F , TF }
  {
    \if_cs_exist:w #1 \cs_end:
      \prg_return_true:
    \else:
      \prg_return_false:
    \fi:
  }
```

(*End definition for* \flag_test:N *and* \flag_test:c. *These functions are documented on page* **??**.)

`\flag_raise:N` `\flag_raise:c` Raising a flag expandably relies on the fact that TEX automatically lets undefined control sequences to `\relax` when building an unknown csname. We build such a csname, then remove it with `\use_none:n`, essentially doing `\use_none:c`, but optimized slightly. When a flag is given as an N-type argument, a little more work is needed, converting to a csname before raising that.

```
\cs_new:Npn \flag_raise:N #1
  { \exp_after:wN \use_none:n \cs:w \cs_to_str:N #1 \cs_end: }
\cs_new:Npn \flag_raise:c #1
  { \exp_after:wN \use_none:n \cs:w #1 \cs_end: }
```

(*End definition for* `\flag_raise:N` *and* `\flag_raise:c`*. These functions are documented on page* ??.)

`\flag_lower:N` `\flag_lower:c` Simply undefine the control sequence, locally.

```
\cs_new_protected:Npn \flag_lower:N #1
  { \cs_set_eq:NN #1 \c_undefined:D }
\cs_generate_variant:Nn \flag_lower:N { c }
```

(*End definition for* `\flag_lower:N` *and* `\flag_lower:c`*. These functions are documented on page* ??.)

```
\msg_kernel_new:nnnn { flag } { already-defined }
  { The~control~sequence~#1~is~already~declared~as~a~flag. }
  {
    LaTeX~was~asked~to~define~the~flag~#1,~but~it~has~already~
    been~defined~as~a~flag.~The~flag~module~is~mostly~meant~
    for~kernel~use,~and~booleans~should~be~preferred.
  }
⟨/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.