Syntax changes in LaTeX3 functions

The LaTeX3 Project*

Released 2019-01-12

This file describes functions that were expected to be completely stable, but whose syntax has changed in ways that may potentially require code relying on them to be changed. This file does not include bug-fixes, nor backward-compatible extensions of the syntax, nor changes to functions in l3candidates, nor functions that were completely deprecated: the latter are listed in `l3obsolete.txt`. Only changes after August 2011 are listed, with an approximate date.

1 August 2011

- `\tl_if_single:n(TF)` recognized any non-zero number of explicit spaces as ⟨*true*⟩, and did not ignore trailing spaces. Now it is ⟨*true*⟩ for

  ⟨*optional spaces*⟩⟨*normal token or brace group*⟩⟨*optional spaces*⟩.

- `\tl_reverse:n` stripped outer braces and lost unprotected spaces. Now it keeps spaces, leaves unbraced single tokens unbraced, and braced groups braced.
- `\tl_trim_spaces:n` only removed one leading and trailing space. Now removes recursively. Also, on the left it used to strip implicit and explicit spaces with any character code. Now it strips only explicit space characters (32, 10).

2 September 2011

- clist functions which receive an `n`-type comma list argument now trim spaces from each item in the argument.

3 May 2012

- The l3fp code has been completely rewritten with a new expandable interface.
- Getting/popping from a comma list or sequence or property list that is empty (or missing the given key) now gives the quark `\q_no_value`.

*E-mail: latex-team@latex-project.org

4 June 2012

- Access to list functions now indexes from 1, not from 0. This applies to multiple choices in the l3keys module and the `\clist_item:Nn`, `\seq_item:Nn` and `\tl_item:Nn` functions.
- `\tl_trim_spaces:n` now requires a variable number of expansions to fully expand, rather than exactly two. Of course, `x`-type expansion still correctly evaluates this function.

5 July 2012

- The `\tl_if_head_eq_meaning:nN`, `\tl_if_head_eq_catcode:nN` and `\tl_if_head_eq_charcode:nN` conditionals now never match when their first argument is empty.

6 August 2012

- `\lua_now:x` is now a standard `x`-type expansion of `\lua_now:n`, which does no expansion. Engine-level expansion is moved to `\lua_now_x:n`, reflecting the fact that this is non-standard in the same way as for example `\str_if_eq_x:nn(TF)`.

7 December 2013

- In l3fp expressions, the badly named functions `round0`, `round-`, `round+` are now named `trunc`, `floor`, `ceil`.

8 May 2014

- Now `\int_step_function:nnnN` evaluates its first three arguments (start, step, stop) up front, rather than evaluating them at each step in the loop. The same holds for the related mappings `\int_step_inline:nnnn`, `\int_step_variable:nnnNn`, and their analogues for `dim` and `fp` datatypes.

9 July 2014

- In l3fp expressions, juxtaposition is interpreted as multiplication. Now the precedence of juxtaposition is set to be the same as if there was an explicit multiplication sign `*`. Previously, juxtaposition would bound more tightly than any other operation.

10 August 2015

- The `\hbox:n` and related l3box commands now take an `n`-type argument and provide it braced to the underlying TEX primitive. The functions `\hbox:w` and `\hbox_end:` in contrast do not read the contents of the box as a macro argument.

11 2016

No change.

12 July 2017

- Boolean expressions are now evaluated eagerly, namely both operands of logical `and` (`&&`) and `or` (`||`) are evaluated even when the result of the logical operation is fixed after determining the first operand. For lazy evaluation, `\bool_lazy_and_p:nn` and related functions are provided.

13 November 2017

- Spaces are now preserved inside keys in l3keys, and trimmed at both ends.
- `\cs_generate_variant:Nn` is now stricter: it only allows to change `N`-type arguments to `c`, and `n` to `o`, `V`, `v`, `f`, `x`. On the one hand the latter argument types typically give rise to more than one token, not suitable for use by an `N`-type base function. On the other hand, `c` variants of `n` arguments should often be `v` variants (when the argument is eventually evaluated) or mistakes where the programmer thought the base function was `N`-type.