The xstacks Package
Version 1.0a

Alceu Frigeri*

January 2026

Abstract

This package aims at solving one problem: how to preserve some variable values (tokens) beyond a variable number of nested groups. That for, four alternative strategies are implemented.

Contents

1 Introduction

Sometimes one needs to preserve the value of some variables beyond a local group. In simple cases it's enough to use `\group_insert_after:N` or `\aftergroup`, if you know how many nested groups you are in.

But, sometimes you don't have this information (see [1], for instance) in which case you have a few options:

- use global variables, or
- implement an after group strategy, as suggested by Carlisle, or
- use a (global) stack.

The global variables way is, of course, the fastest (and easiest) if you don't have to worry about reentrant coding (like when you have nested groups inside an environment, which might get nested into itself).

`\xstacks_aftergroup:N` and `\xstacks_aftergroup:NN` (and associated) are two variants of the after group strategy.

`\xstacks_cs_gset:N` and `\xstacks_gset:N` (and associated) are two variants of the (global) stack strategy.

*https://github.com/alceu-frigeri/xstacks

2 Simple mark point after group variant

`\xstacks_groupmark:` `\xstacks_groupmark:`
`\xstacks_aftergroup:N` `\xstacks_aftergroup:N` {⟨token⟩}

These will use a single, internal, variable to 'track' the target group level. Better said, upon calling `\xstacks_groupmark:` the current group will be saved (local assignment), so that, later on, `\xstacks_aftergroup:N` can be called from nested groups and ⟨token⟩ will be pushed into the marked group.

> ***Note:*** Since all assignments are local, it's possible to have multiple marks, for instance, one at group level 2, another at group level 5, so that anything saved with `\xstacks_aftergroup:N` on group level 6+ will be restored at group level 5... anything between level 3 until the other mark will be restored at level 2.
>
> ***Note:*** if `\xstacks_aftergroup:N` is called at the same level (or above) of the mark, it will be equivalent to a simple `\group_insert_after:N`.

3 Multiple mark points after group variant

`\xstacks_groupmark:N` `\xstacks_groupmark:N` {⟨int-var⟩}
`\xstacks_aftergroup:NN` `\xstacks_aftergroup:NN` {⟨int-var⟩} {⟨token⟩}

⟨int-var⟩ must be an already declared integer variable, and will be used to mark/track a group level. That way it is possible to have multiple and independent return points. Otherwise it works exactly as the previous pair of commands. All assignments made to ⟨int-var⟩ are also local.

4 Stack command variant

`\xstacks_cs_gset:N` `\xstacks_cs_gset:N` {⟨cs-radix⟩}

This will globally create a set of commands, named after ⟨cs-radix⟩, to push, put and pop items from a private global stack. All assignments to/from that stack will be global, and the stack itself will be unique to the command's set.

> ***Note:*** It won't test for the existence of ⟨cs-radix⟩ and silently overwrite any previous definition.

`\<cs-radix>_gpush:n` `\<cs-radix>_gpush:n` {⟨tokens⟩}
`\<cs-radix>_gput_right:n` `\<cs-radix>_gput_right:n` {⟨tokens⟩}
`\<cs-radix>_gput_left:n` `\<cs-radix>_gput_left:n` {⟨tokens⟩}
`\<cs-radix>_gpop:` `\<cs-radix>_gput_gpop:`

The `\<cs-radix>_gpush:n` will push ⟨tokens⟩ (can be any number of tokens) into a global, private, stack. `\<cs-radix>_gput_right:n` and `\<cs-radix>_gput_left:n` will amend tokens to it, and `\<cs-radix>_gpop:`, as the name implies, will insert the top of the stack into the input stream. That way it is possible to have a very fine control of what, where and when the items are collected and used.

5 Stack variable variant

`\xstacks_gset:N` `\xstacks_gset:N` {⟨stack-var⟩}
`\xstacks_gpush:Nn` `\xstacks_gpush:Nn` {⟨stack-var⟩} {⟨tokens⟩}
`\xstacks_gput_right:Nn` `\xstacks_gput_right:Nn` {⟨stack-var⟩} {⟨tokens⟩}
`\xstacks_gput_left:Nn` `\xstacks_gput_left:Nn` {⟨stack-var⟩} {⟨tokens⟩}
`\xstacks_gpop:N` `\xstacks_gpop:N` {⟨stack-var⟩}

`\xstacks_gset:N` will globally create a stack variable named ⟨stack-var⟩ (a specialized token list variable). Once created it is possible to push tokens into it (`\xstacks_gpush:Nn`), amend tokens to the top (`\xstacks_gput_right:Nn` and `\xstacks_gput_left:Nn`) and pop those tokens (`\xstacks_gpop:N`) into the input stream. All assignments being global.

> ***Note:*** It won't test for the existence of ⟨stack-var⟩ and silently overwrite any previous definition.

6 Benchmarks and Final Thoughts

In the following, there is an exercise of ..., better said, to evaluate the advantage/disadvantage of each approach, in an extreme case: multiple tokens, deeply nested groups. The most effective strategy still is the after group one, a difference of just about *200 ops*, but, taking in account that the stack variants are mostly 'constant time' (they don't depend on how deep the grouping is, but just how many operations (push/pop) are needed), in lighter cases there is no content.

At the end, it's a case of flexibility/convenience versus performance. In case of the original problem, to save context past the end of a scope, the after group approach is the one. Now, if one needs a finer control of what/when (and not just preserving context past end of scope) the stack variants can have some use.

> ***Note:*** The after group variant is the fastest if using the `\if_int_compare:w` primitive. If instead, for reference, `\int_if_compare:nNnTF` is used, the number of *ops* almost triple! Take a look at the code, the version with `\int_if_compare:nNnTF` is commented out.
>
> ***Note:*** Not really needed, but since one is at it, you can try (for more stable results) `\fp_set:Nn \g_benchmark_duration_target_fp {20}`...

6.1 Single mark, after group variant

```
\benchmark:n
  {
    \group_begin:
      \xstacks_groupmark:
    {{
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpc_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpc_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpc_bool
    {{
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpb_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpb_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpb_bool
    {{
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
    {{
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
    {{
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
        \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
    }} }} }}
    }} }}
    \group_end:
  }
```

On average, it took about *160 ops* (*430 ops* if using `\int_if_compare:nNnTF`). If 'only' the 4 first groups, the average goes down to just *35 ops* (*90 ops* if using `\int_if_compare:nNnTF`). Obviously, the number of after groups raises exponentially, 2^n, with the number of nested groups.

6.2 Multiple marks, after group variant

```
    \int_gzero_new:N \myMark_int
\benchmark:n
  {
    \group_begin:
      \xstacks_groupmark:N \myMark_int
    {{
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpc_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpc_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpc_bool
    {{
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpb_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpb_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpb_bool
    {{
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
    {{
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
    {{
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
        \xstacks_aftergroup:NN \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN \myMark_int \l__mytest_tmpa_bool
    }} }} }}
    }} }}
    \group_end:
  }
```

On average, it took about *200 ops* (*480 ops* if using `\int_if_compare:nNnTF`). If 'only' the 4 first groups, the average goes down to just *45 ops* (*95 ops* if using `\int_if_compare:nNnTF`). Likewise, the number of after groups raises exponentially, 2^n, with the number of nested groups (more expensive than the previous one because of the extra integer that has to be carried on).

6.3 Stack command variant

```
    \xstacks_cs_gset:N {myStack}
\benchmark:n
  {
    \group_begin:
    {{
        \myStack_gpush:n {
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
        }
    {{
        \myStack_gput_right:n {
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
        }
    {{
        \myStack_gput_right:n {
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
        }
    {{
        \myStack_gput_right:n {
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
        }
    {{
        \myStack_gput_right:n {
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
        }
    }} }} }}
    }} }}
      \myStack_gpop:
    \group_end:
  }
```

On average, it took about *405 ops*. If 'only' the 4 first groups, the average goes down to about *195 ops*. All operations, `\myStack_gpush:n`, `\myStack_gput_right:n` and `\myStack_gpop:` are, more or less, equally expensive.

6.4 Stack variable variant

```
    \xstacks_gset:N \g_mytest_stack
\benchmark:n
  {
    \group_begin:
    {{
        \xstacks_gpush:Nn \g_mytest_stack {
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
        }
    {{
        \xstacks_gput_right:Nn \g_mytest_stack {
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
          \bool_set_true:N \l__mytest_tmpb_bool
        }
    {{
        \xstacks_gput_right:Nn \g_mytest_stack {
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
        }
    {{
        \xstacks_gput_right:Nn \g_mytest_stack {
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
        }
    {{
        \xstacks_gput_right:Nn \g_mytest_stack {
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
          \bool_set_true:N \l__mytest_tmpc_bool
        }
    }} }} }}
    }} }}
      \xstacks_gpop:N   \g_mytest_stack
    \group_end:
  }
```

On average, it took about *310 ops*. If 'only' the 4 first groups, the average remains about *290 ops*. The `\xstacks_gpush:Nn` and `\xstacks_gpop:N` are the most expensive operations in this case.

References

[1] David Carlisle. *Stackexchange about grouping*. 2026. URL: https://tex.stackexchange.com/questions/757755/coffins-scope-groups#comment1889872_757755 (visited on 01/01/2026).

7 Addendum

A quick test ...

```
\ExplSyntaxOn
  \bool_new:N \l__mytest_tmpa_bool
  \bool_new:N \l__mytest_tmpb_bool
  \bool_new:N \l__mytest_tmpc_bool
  \cs_new:Npn \mytest_show_bools:n #1
    { \underline{#1:}\par
      \bool_if:NTF \l__mytest_tmpa_bool {{\color{red}a~true}}{a~false}  \par
      \bool_if:NTF \l__mytest_tmpb_bool {{\color{red}b~true}}{b~false}  \par
      \bool_if:NTF \l__mytest_tmpc_bool {{\color{red}c~true}}{c~false}  \par
    }
\ExplSyntaxOff
```

```
\ExplSyntaxOn
    \group_begin:
      \xstacks_groupmark:
    {{  \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpa_bool
    {{  \xstacks_aftergroup:N \bool_set_true:N
        \xstacks_aftergroup:N \l__mytest_tmpb_bool
        \mytest_show_bools:n {T1}
    }}}}
      \mytest_show_bools:n {T2}
    \group_end:
\ExplSyntaxOff
```

T1:
a false
b false
c false
T2:
a true
b true
c false

```
\ExplSyntaxOn
    \int_new:N \myMark_int
    \group_begin:
      \xstacks_groupmark:N \myMark_int
    {{  \xstacks_aftergroup:NN
          \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN
          \myMark_int \l__mytest_tmpa_bool
    {{  \xstacks_aftergroup:NN
          \myMark_int \bool_set_true:N
        \xstacks_aftergroup:NN
          \myMark_int \l__mytest_tmpb_bool
        \mytest_show_bools:n {T1}
    }}}}
      \mytest_show_bools:n {T2}
    \group_end:
\ExplSyntaxOff
```

T1:
a false
b false
c false
T2:
a true
b true
c false

```
\ExplSyntaxOn
    \group_begin:
      \xstacks_cs_gset:N {myStack}
    {{  \myStack_gpush:n
          {\bool_set_true:N \l__mytest_tmpa_bool}
    {{  \myStack_gput_right:n
          {\bool_set_true:N \l__mytest_tmpb_bool}
        \mytest_show_bools:n {T1}
    }}}}
      \myStack_gpop:
      \mytest_show_bools:n {T2}
    \group_end:
\ExplSyntaxOff
```

T1:
a false
b false
c false
T2:
a true
b true
c false

```
\ExplSyntaxOn
    \group_begin:
      \xstacks_gset:N \g_mytest_stack
    {{  \xstacks_gpush:Nn \g_mytest_stack
          {\bool_set_true:N \l__mytest_tmpa_bool}
    {{  \xstacks_gput_right:Nn \g_mytest_stack
          {\bool_set_true:N \l__mytest_tmpb_bool}
        \mytest_show_bools:n {T1}
    }}}}
      \xstacks_gpop:N   \g_mytest_stack
      \mytest_show_bools:n {T2}
    \group_end:
\ExplSyntaxOff
```

T1:
a false
b false
c false
T2:
a true
b true
c false