`multiexpand`
Trigger multiple expansions in one expansion step*

Bruno Le Floch[†‡]

Released 2013/01/08

Contents

1 Two user commands

- For $n > 0$, expanding `\MultiExpand{`n`}\macro` twice gives the n-th expansion of `\macro`.
- For $n > 0$, expanding `\MultiExpandAfter{`n`}\macroA\macroB` twice expands `\macroB` n times before expanding `\macroA`.

Note that neither functions work for $n = 0$.

These can typically be combined as

```
\MultiExpand{7}%
```

*This file describes version 1.1, last revised 2013/01/08.

[†]E-mail: blflatex@gmail.com

[‡]I have gathered ideas from various posts in the `{TeX}` community at `http://tex.stackexchange.com`. Thanks to their authors.

```
\MultiExpandAfter{4}\a\MultiExpandAfter{7}\b%
\MultiExpandAfter{3}\c\d
```

which would expand `\d` 3 times, then `\c` 5 times (2 of the 7 times were used to expand `\MultiExpandAfter{3}`), then `\b` twice ($4-2$), and finally `\a` five times ($7-2$). Note that all this happens in precisely *two* steps of expansion.

In some cases, one needs to achieve the same effect in *one* step only. For this, we use the first expansion of `\MultiExpand`, which is `\romannumeral \multiexpand`, or of `\MultiExpandAfter`, which is `\romannumeral \multiexpandafter`). Expanding `\romannumeral \multiexpand{`n`}` once expands the following token n times, and similarly for `\romannumeral \multiexpandafter{`n`}`.

These are especially useful when we want to expand several times a very specific token which is buried behind many others. Example: expanding the following code twice,

```
\MultiExpand{3}\expandafter\macroA\expandafter\macroB%
\romannumeral\multiexpandafter{4}\macroC\macroD
```

will expand `\macroD` 4 times, then will expand `\macroA` $2 = 3-1$ additional times.

Note: as we mentionned, this breaks for $n = 0$. But in this case, consider using `\expandafter\empty`, or a variant thereof.

2 How it works

The primitive `\romannumeral` expands what follows fully until it builds a full number. It will remove exactly one trailing space if the first non-digit token is a space. So if we expand the construction `\romannumeral0\expandafter\space` once, then `\romannumeral` will see the `0`, and expand `\expandafter`: it cannot yet be sure that it won't find another digit afterwards. This expands the next token once. In other words, `\romannumeral0\expandafter\space` behaves as if it was not there.

This is how we end our loop. Namely, `\multiexpand{`$\langle n \rangle$`}` checks if $n < 2$, in which case it stops with `0\expandafter\space`. If $n \geq 2$, then it simply expands to `\multiexpand{`$\langle n-1 \rangle$`}`, plus the relevant `\expandafter`s meant to expand the next token once.[1]

[1]Note to self: Possible optimization: put three `\expandafter` rather than one at the

3 Implementation

```
1 ⟨*package⟩
```

We work inside a group, to change the catcode of `@`. So we will only do `\gdef`s. We also define a macro `\ME@use`.

```
2 \begingroup
3 \catcode`\@=11\relax%
4 \gdef\ME@use#1{#1}%
```

`\MultiExpand` and `\MultiExpandAfter` are simply shorthands to avoid typing `\romannumeral`. Drawback: they require two steps of expansion rather than only one.

```
5 \gdef\MultiExpand{\romannumeral\multiexpand}%
6 \gdef\MultiExpandAfter{\romannumeral\multiexpandafter}%
```

The `\romannumeral` is stopped by `0\space`. We insert the relevant `\expandafter` to do the work.

```
7 \xdef\ME@endroman#1{0\noexpand\expandafter\space}%
8 \xdef\ME@endroman@after#1{0\noexpand\expandafter\space\noexpand\expandafter}%
9 \long\gdef\multiexpand#1{%
10   \ifnum#1<2 \expandafter \ME@endroman%
11   \else      \expandafter \ME@use%
12   \fi%
13   {\expandafter \multiexpand \expandafter {%
14       \number\numexpr#1-1\expandafter}}%
15 }%
```

Almost identical definitions for expanding after...

```
16 \long\gdef\multiexpandafter#1{%
17   \ifnum#1<2 \expandafter\ME@endroman@after%
18   \else      \expandafter \ME@use%
19   \fi%
20   {\expandafter \multiexpandafter \expandafter {%
21       \number\numexpr#1-1\expandafter}\expandafter}%
22 }%
```

Close the group.

```
23 \endgroup
```

```
24 ⟨/package⟩
```

---

end, to to two expansions at once.