Update marks in box*

WenJian Chern†

March 8, 2025 v0.3

Abstract

There are some cases where one may put title in box and still want its `\markboth` working. This brings the `updatemarks` package. The `updatemarks` package provides interface to extract marks where are in a inner box and then can put the first and last marks back to outer. It can automatically update marks where are in `minipage`, *boxed multicols* and `tcolorbox` environments.

Contents

1 Usage of the package

You simply insert `\usepackage{updatemarks}` or `\usepackage[`⟨*options*⟩`]{updatemarks}` in the preamble of your document.

There are three package options `minipage`, `multicol`(an optional `s`) and `tcolorbox`, which are used to enable the function of automatical updating marks in `minipage`, `multicols` and `tcolorbox` (both `tcolorbox` environment and `\tcbox` command), respectively.

No automatical updatings are enable, by default.

If your LaTeX version is 2022-06-01 or newer, you can use `\SetKeys[updatemarks]{`⟨*options*⟩`}` to enable or disable automatical updating locally.

Such as,

```
\SetKeys[updatemarks]{minipage=false, tcolorbox=true, multicol}
```

*Issues: https://github.com/Sophanatprime/updatemarks/issues

†Email: longaster@163.com

so marks in `minipage` would not update, whereas `tcolorbox` and `multicols` do.

The total empty mark will not be extracted and updated. But `\markboth{}{}` and `\InsertMark{}` do will be extracted and updated, because they are not total empty internally.

However, the updating is not once for all. For example, if you are using `minipage` in `\makebox`, even if you enable the automatical updating, the marks still can not be found by LaTeX. But, if you put `minipage` in `tcolorbox` or the other way round, LaTeX still can get correct marks.

The point is, updating marks is done inner to outer and level by level, if any level is not updated, you will lost all marks in inner boxes. Furthermore, whether marks are updated or not is only influenced by box level instead of group level, so if you try to limit marks in a certain position, you should use a box instead of `\begingroup` and `\endgroup`.

The extracting and updating would not remove these marks, they still be there, you can find them by your own method as well.

2 `minipage`

You can enable automatically extracting and updating marks where are in `minipage` to its nearest outer box level, by using `minipage` package option or set `minipage` key to true.

updatemarks patches the `\endminipage`, if you enable the function, marks in any other environments that use `minipage` will be updated, including `varwidth` of varwidth package, etc.

3 Version 2.* of multicol

multicol has fully supported the new mark mechanism of LaTeX since version 2.0, so no more things are needed to be done by updatemarks.

4 Version 1.* of multicol and adjmulticol

If `multicols` or `adjmulticols` or their starred versions are put in a box — so called *boxed multicols*, you are able to eanble automatically extracting and updating marks into the box (not outer of the box, because they are only updated one level). If you need to update marks in this box, you have to do it manually. Of cause, if the box is `minipage` or `tcolorbox` or another `multicols` or any other supported environments or commands, then no more things to do.

All `multicols` and `adjmulticols` and their starred versions share the same option `multicol`(an optional `s`).

For *non boxed multicols*, i.e, not in any other box except the main vertical list, the `\topmark` (and `\topmarks`) at first and last page are not correct. Multiple contiguous forced break or `\clearpage` may cause the wrong marks.

If your LaTeX version is 2022-06-01 or newer, you can get more correct mark values by using `\TopMark`, `\FirstMark` and `\LastMark` in head and foot, but only the `previous-page` region and `page` region are supported for now. Still, Multiple contiguous forced break or `\clearpage` may cause the wrong marks.

5 tcolorbox

The updatemarks package can also update marks in `tcolorbox` environments (both breakable and unbreakable) and `\tcbox` commands.

You are allowed to use `updatemarks=true` or `updatemarks` in `\tcbset` or as environment and command option to enable the function, and `updatemarks=false` to disable.

6 Set automatical updating list

updatemarks can automatically detect mark classes allocated by `\newmarks` and `\NewMarkClass` in preamble. But if you allocate mark class after preamble (which I strongly recommend you do not), updatemarks also provides interfaces to enable you add these mark class to automatical updating list. If you are not using these two commands, then nothing need to be done.

`\AddToUpdateMarksList` `\SetUpdateMarksList` `\RemoveFromUpdateMarksList`

```
\AddToUpdateMarksList      {⟨number list⟩}
\SetUpdateMarksList        {⟨number list⟩}
\RemoveFromUpdateMarksList {⟨number list⟩}
```

You can use these functions to globally add items to, set items, and remove items from the list of mark classes which need to be automatically updated.

The ⟨*number*⟩ is the first argument of `\newmarks`, or literally, a number. Specially, for those mark classes declared by `\NewMarkClass`, you are able to use `[`⟨*class*⟩`]` or `\MarkClass{`⟨*class*⟩`}`. Such as

```
\RemoveFromUpdateMarksList { [2e-right], 4, \MarkClass{my-class} }
```

You can also use number ranges in ⟨*number list*⟩, such as `2 -> 5`.

`\AddAllocatedToUpdateMarksList`

```
\AddAllocatedToUpdateMarksList
```

The function add all mark classes allocated by `\newmarks` and `\NewMarkClass` to automatical updating list.

7 Generic interfaces of extracting and updating

If you need to extract and update marks manually, `\ExtractMarks`, `\ExtractSplitMarks` and `\UpdateMarks` will help you.

`\ExtractMarks`

```
\ExtractMarks {⟨box number⟩}
\ExtractMarks [⟨number list⟩] {⟨box number⟩}
\ExtractMarks * {⟨text⟩}
\ExtractMarks * [⟨number list⟩] {⟨text⟩}
```

Save first marks and last marks of specified mark classes which are directly presented in ⟨*box number*⟩ or ⟨*text*⟩ if these marks is not total empty. Marks in deeper boxes will not be detected, unless they are moved out.

If ⟨*number list*⟩ is presented, the mark classes are these numbers, otherwise they are the automatical updating list.

The ⟨*box number*⟩ is the first argument of `\newbox` or `\newsavebox`, etc., and have saved contents by using `lrbox` or `\sbox` or similar.

The ⟨*text*⟩ is typeset material and *will be executed*.

If the ⟨*box number*⟩ is a vbox and contains forced page break, then all marks after the first forced page break will not be detected.

`\ExtractSplitMarks`

```
\ExtractSplitMarks
\ExtractSplitMarks [⟨number list⟩]
```

This command is used after a `\vsplit`, and save first marks and last marks of specified mark classes which are directly presented in being split part if these marks is not total empty. Marks in deeper boxes will not be detected, unless they are moved out.

If ⟨*number list*⟩ is presented, the mark classes are these numbers, otherwise they are the automatical updating list.

`\UpdateMarks`

```
\UpdateMarks
\UpdateMarks [⟨number list⟩]
```

Reinserting saved first marks and last marks of specified mark classes (if have been saved) into the current box, or if not in a box then the main vertical list.

If ⟨*number list*⟩ is presented, the mark classes are these numbers, otherwise they are the automatical updating list.

`insertmark` hook would not be used as it's already been used.

\ExtractMarksTo
New: 2024-02-19

```
\ExtractMarksTo {⟨box number⟩} {⟨cmd⟩}
\ExtractMarksTo [⟨number list⟩] {⟨box number⟩} {⟨cmd⟩}
\ExtractMarksTo * {⟨text⟩} {⟨cmd⟩}
\ExtractMarksTo * [⟨number list⟩] {⟨text⟩} {⟨cmd⟩}
```

It collects all marks to ⟨*cmd*⟩, then you can use ⟨*cmd*⟩ to reinsert these marks.

`insertmark` hook would not be used as it's already been used.

8 Disable patches or write your own patches

By default, updatemarks use its own patches to support `minipage`, `multicols` and `tcolorbox`, however, you can write your own patches and remove the patches will be done by updatemarks.

To the patches for `minipage`, you can define `\updatemarks@minipage@patch` before updatemarks is loaded, then updatemarks will use your patches. Specially, if you set `\updatemarks@minipage@patch` to empty, then no patches will be done for `\endminipage`.

The patches for multicol, adjmulticol and tcolorbox, are `\updatemarks@multicol@patch`, `\updatemarks@adjmulticol@patch`, `\updatemarks@tcolorbox@patch` and `\updatemarks@multicolnewmark@patch`, `\updatemarks@adjmulticolnewmark@patch` for new mark mechanism, you are able set them in preamble before or after updatemarks is loaded.

9 Programming interfaces

This section describes the interfaces of LaTeX3.

\updatemarks_parse_classes:Nn
New: 2024-02-19

```
\updatemarks_parse_classes:Nn ⟨seq var⟩ {⟨number list⟩}
```

Parse number list and save it to ⟨*seq var*⟩.

\updatemarks_extract:nN
\updatemarks_extract_split:N

```
\updatemarks_extract:nN {⟨material⟩} ⟨seq var⟩
\updatemarks_extract_split:N ⟨seq var⟩
```

Programming interface of `\ExtractMarks` and `\ExtractSplitMarks`, respectively.

⟨*material*⟩ is content to build a box. Such as unpacked box using `\hbox_unpack:N` or `\vbox_unpack:N` or text.

⟨*seq var*⟩ is the number sequence of mark classes which are need to be extracted.

They only save the marks whose positions are first or last.

\updatemarks_extract:nNN
New: 2024-02-19

```
\updatemarks_extract:nNN {⟨material⟩} ⟨seq var⟩ ⟨tl var⟩
```

Programming interface of `\ExtractMarksTo`.

`insertmark` hook would not be used as it's already been used.

\updatemarks_extract_act:nNn
New: 2024-02-19

```
\updatemarks_extract_act:nNn {⟨material⟩} ⟨seq var⟩ {⟨code⟩}
```

Run ⟨*code*⟩ for every mark classes in ⟨*seq var*⟩, the code can use one parameter, which is the current mark class, and two `tl` variable `\l_updatemarks_first_tl` and `\l_updatemarks_last_tl`, which save the first and last marks at specified mark class, respectively. If these two `tl` is not exist, then no marks at the mark class are inserted.

\updatemarks_update:N

```
\updatemarks_update:N ⟨seq var⟩
```

This function has the same function of `\UpdateMarks`.

⟨*seq var*⟩ is the number sequence of mark classes which are need to be reinserted into the current box or the main vertical list.

It only reinserts marks whose positions are first or last.

`insertmark` hook would not be used as it's already been used.

\updatemarks_save:Nnn `\updatemarks_save:Nnn ⟨seq var⟩ {⟨position⟩} {⟨value code⟩}`

Saving marks at ⟨*position*⟩ of specified mark classes ⟨*seq var*⟩, whose values are expanding ⟨*value code*⟩ *once*.

If `\l_updatemarks_nonempty_bool` is set to `true`, then the expanded value will not be saved if it is total empty.

The ⟨*value code*⟩ receives each item in the ⟨*seq var*⟩ as a trailing brace group, then expand the whole part of ⟨*value code*⟩ and the trailing group *once*.

\updatemarks_save_e:Nnn `\updatemarks_save_e:Nnn ⟨seq var⟩ {⟨position⟩} {⟨value code⟩}`

Saving marks at ⟨*position*⟩ of specified mark classes ⟨*seq var*⟩, whose values are fully expanding ⟨*value code*⟩.

If `\l_updatemarks_nonempty_bool` is set to `true`, then the expanded value will not be saved if it is total empty.

The ⟨*value code*⟩ receives each item in the ⟨*seq var*⟩ as a trailing brace group, then *fully expand* the whole part of ⟨*value code*⟩ and the trailing group.

\updatemarks_alias:Nnn `\updatemarks_alias:Nnn ⟨seq var⟩ {⟨alias position⟩} {⟨source position⟩}`

Setting marks at ⟨*alias position*⟩ are equal to ⟨*source position*⟩ of specified mark classes ⟨*seq var*⟩.

If `\l_updatemarks_nonempty_bool` is set to `true`, then the expanded value will not be saved if it is total empty.

\updatemarks_remove:Nn `\updatemarks_remove:Nn ⟨seq var⟩ {⟨position⟩}`

Remove marks at ⟨*position*⟩ of specified mark classes ⟨*seq var*⟩.

\updatemarks_value:nn ⋆ `\updatemarks_value:nn {⟨position⟩} {⟨mark class⟩}`

Return the value of ⟨*mark class*⟩ at ⟨*position*⟩. If it has not been saved, then return to total empty.

TEXhackers note: The result is returned within the `\unexpanded` primitive (`\exp_not:n`).

\g_updatemarks_max_int Readonly interger, its value is the maximum mark class number allocated.

\g_updatemarks_seq The number sequence which holds the mark classes needed to be automatical updated.

\g_updatemarks_classes_seq Readonly number sequence which holds the mark classes allocated by `\NewMarkClass`. If `\NewMarkClass` is undefined, then it's empty.

\l_updatemarks_nonempty_bool

A local `bool` variable which is used to control if is going to save a total empty value.

\l_updatemarks_first_tl
\l_updatemarks_last_tl A local `tl` variable which are use to save the first and last marks.

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.