

The `postnotes` package*

Code documentation

Gustavo Barros[†]

2023-08-21

Contents

1	Initial setup	2
2	Data	2
3	Options	5
4	<code>\postnote</code>	10
5	<code>\postnoteref</code>	16
6	<code>\postnotesection</code>	17
7	<code>\printpostnotes</code>	18
8	Headers	24
9	Compatibility	30
10	Languages	38
	Index	41

*This file describes v0.2.6, released 2023-08-21.

[†]<https://github.com/gusbrs/postnotes>

1 Initial setup

Start the DocStrip guards.

```
1 <*package>
   Identify the internal prefix (LATEX3 DocStrip convention).
2 <@@=postnotes>
```

The new syntax for file/package hooks, which the package assumes, requires kernel 2021-11-15 (ltnnews34, ltfilehook). Furthermore, the kernel of 2022-06-01 introduced a couple of very nice features which simplifies the relation with hyperref (ltnnews35, hyperref-linktarget): the provision of `\MakeLinkTarget` and the definition by the kernel of the starred version of `\ref`, which we can use regardless of hyperref being loaded. So we require the 2022-06-01 kernel or newer.

```
3 \def\postnotes@required@kernel{2022-06-01}
4 \NeedsTeXFormat{LaTeX2e}[\postnotes@required@kernel]
5 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
6 \IfFormatAtLeastTF{\postnotes@required@kernel}
7   {}
8   {%
9     \PackageError{postnotes}{LaTeX kernel too old}
10    {%
11      'postnotes' requires a LaTeX kernel \postnotes@required@kernel\space or newer.%
12    }%
13  }%
14 \ProvidesExplPackage {postnotes} {2023-08-21} {0.2.6}
15 {Endnotes for LaTeX}
```

2 Data

`__postnotes_data_name:n` Returns the name of the property list variable which stores the data of the `\postnote` with `<note id>` number.

```
   \__postnotes_data_name:n {<note id>}
16 \cs_new:Npn \__postnotes_data_name:n #1
17   { g__postnotes_ #1 _data_prop }
18 \cs_generate_variant:Nn \__postnotes_data_name:n { e }
```

(End of definition for `__postnotes_data_name:n`.)

`postnotes` provides a number of hooks from the new hook system to grant some points of access in key places of the package. Note that hooks created with `\NewHook` are meant to be public interfaces (see <https://chat.stackexchange.com/transcript/message/62955941#62955941>, and following discussion).

`__postnotes_store:mn` Stores the metadata and `<note content>` of `\postnote` with ID `<note id>`, from where it is called. The `postnotes/note/store` hook is intended to add further data to the note, when required to support packages with specific needs.

```
   \__postnotes_store:mn {<note id>} {<note content>}
```

```

19 \NewHook { postnotes/note/store }
20 \cs_new_protected:Npn \__postnotes_store:nn #1#2
21 {
22   \prop_new:c { \__postnotes_data_name:e {#1} }
23   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { note }
24   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { mark }
25     { \l__postnotes_mark_tl }
26   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { counter }
27     { \int_use:N \c@postnote }
28   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { sortnum }
29     {
30       \bool_if:NTF \l__postnotes_manual_sortnum_bool
31         { \fp_use:N \l__postnotes_sort_num_fp }
32         { \int_use:N \c@postnote }
33     }
34   \cs_if_exist:cT { chapter }
35     {
36       \prop_gput:cnx { \__postnotes_data_name:e {#1} }
37         { thechapter } { \thechapter }
38     }
39   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { thesection }
40     { \thesection }
41   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { pnsectname }
42     { \g__postnotes_section_name_tl }
43   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { pnsectid }
44     { \int_use:N \g__postnotes_sectid_int }
45   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { multibool }
46     { \bool_to_str:N \l__postnotes_maybe_multi_bool }
47   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}
48   \UseHook { postnotes/note/store }
49 }

```

(End of definition for __postnotes_store:nn.)

__postnotes_store_section:nn Stores the metadata and *<note content>* of \postnotessection with ID *<note id>*, from where it is called.

```

    \__postnotes_store_section:nn {<note id>} {<note content>}
50 \cs_new_protected:Npn \__postnotes_store_section:nn #1#2
51 {
52   \prop_new:c { \__postnotes_data_name:e {#1} }
53   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { type } { section }
54   \cs_if_exist:cT { chapter }
55     {
56       \prop_gput:cnx { \__postnotes_data_name:e {#1} }
57         { thechapter } { \thechapter }
58     }
59   \prop_gput:cnx { \__postnotes_data_name:e {#1} } { thesection }
60     { \thesection }
61   \prop_gput:cnn { \__postnotes_data_name:e {#1} } { content } {#2}
62 }
63 \cs_generate_variant:Nn \__postnotes_store_section:nn { nx }

```

(End of definition for __postnotes_store_section:nn.)

`__postnotes_prop_get:nnN` Convenience functions to retrieve and clear data from a note based on the ID number.

`__postnotes_prop_item:nn`
`__postnotes_prop_gclear:n`

```

\__postnotes_prop_get:nnN {<note id>} {<property>} {<tl var to set>}
\__postnotes_prop_item:nn {<note id>} {<property>}
\__postnotes_prop_gclear:n {<note id>}

```

```

64 \cs_new_protected:Npn \__postnotes_prop_get:nnN #1#2#3
65 {
66   \prop_get:cnNF { \__postnotes_data_name:e {#1} } {#2} #3
67   { \tl_clear:N #3 }
68 }
69 \cs_new:Npn \__postnotes_prop_item:nn #1#2
70 { \prop_item:cn { \__postnotes_data_name:e {#1} } {#2} }
71 \cs_new_protected:Npn \__postnotes_prop_gclear:n #1
72 { \prop_gclear:c { \__postnotes_data_name:e {#1} } }

```

(End of definition for `__postnotes_prop_get:nnN`, `__postnotes_prop_item:nn`, and `__postnotes_prop_gclear:n`.)

`\post@note` The `\newlabel` equivalent for postnotes. Based on the kernel's `\@newl@bel` so that we get L^AT_EX checks for multiple and changed references for free (procedure learnt from `zref`). `\post@note`, when the `.aux` file is read, defines macros named `\postnote@r@<label name>`, according to the prefix set by `\c__postnotes_ref_prefix_tl`.

```

\post@note {<label name>} {<label content>}

```

```

73 \tl_const:Nn \c__postnotes_ref_prefix_tl { postnote@r }
74 \cs_new_protected:Npx \post@note #1#2
75 { \exp_not:N \@newl@bel { \c__postnotes_ref_prefix_tl } {#1} {#2} }

```

(End of definition for `\post@note`.)

And ensure `\post@note` is defined in the `.aux` file. The hooks are the same used by `hyperref` for similar purpose.

```

76 \AddToHook { begindocument }
77 {
78   \legacy_if:nT { @filesw }
79   {
80     \iow_now:Nx \@mainaux
81     { \token_to_str:N \providecommand \token_to_str:N \post@note [2]{ } }
82   }
83 }
84 \AddToHook { include/before }
85 {
86   \legacy_if:nT { @filesw }
87   {
88     \iow_now:Nx \@partaux
89     { \token_to_str:N \providecommand \token_to_str:N \post@note [2]{ } }
90   }
91 }

```

`__postnotes_set_label:nn` Label setting functions for each pertinent context. They must use `\iow_shipout_x:Nn`, since the main information we are interested in is the page.

`__postnotes_set_mark_page_label:n`
`__postnotes_set_text_page_label:n`
`__postnotes_set_print_page_label:n`

```

    \__postnotes_set_label:nn {<label name>} {<value>}
    \__postnotes_set_mark_page_label:n {<note id>}
    \__postnotes_set_text_page_label:n {<note id>}
    \__postnotes_set_print_page_label:n {<note id>}
92 \cs_new_protected:Npn \__postnotes_set_label:nn #1#2
93 {
94   \legacy_if:nT { @filesw }
95   {
96     \iow_shipout_x:Nn \@auxout
97     { \token_to_str:N \post@note { #1 } { #2 } }
98   }
99 }
100 \cs_new_protected:Npn \__postnotes_set_mark_page_label:n #1
101 { \__postnotes_set_label:nn { mark@ #1 } { \thepage } }
102 \cs_generate_variant:Nn \__postnotes_set_mark_page_label:n { x }
103 \cs_new_protected:Npn \__postnotes_set_text_page_label:n #1
104 { \__postnotes_set_label:nn { text@ #1 } { \int_use:N \c@page } }
105 \cs_generate_variant:Nn \__postnotes_set_text_page_label:n { x }
106 \cs_new_protected:Npn \__postnotes_set_print_page_label:n #1
107 { \__postnotes_set_label:nn { print@ #1 } { \int_use:N \c@page } }
108 \cs_generate_variant:Nn \__postnotes_set_print_page_label:n { x }

```

(End of definition for __postnotes_set_label:nn and others.)

__postnotes_get_pageref:Nn Reference data extraction functions.
 __postnotes_extract_pageref:n

```

    \__postnotes_get_pageref:Nn {<tl var to set>} {<label name>}
    \__postnotes_extract_pageref:n {<label name>}
109 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
110 {
111   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
112   { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
113   { \tl_clear:N #1 }
114 }
115 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Nx }
116 \cs_new:Npn \__postnotes_extract_pageref:n #1
117 {
118   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
119   { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
120   { \c_empty_tl }
121 }
122 \cs_generate_variant:Nn \__postnotes_extract_pageref:n { e }

```

(End of definition for __postnotes_get_pageref:Nn and __postnotes_extract_pageref:n.)

3 Options

heading option

```

123 \keys_define:nn { postnotes/setup }
124 {
125   heading .cs_set_protected:Np = \pnheading ,

```

```

126     heading .value_required:n = true ,
127   }

```

`\pnheading` Provide default value for `\pnheading`.

```

128 \cs_if_exist:cTF { chapter }
129 {
130   \cs_new_protected:Npn \pnheading
131     {
132       \chapter*{\pntitle}
133       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
134     }
135 }
136 {
137   \cs_new_protected:Npn \pnheading
138     {
139       \section*{\pntitle}
140       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
141     }
142 }

```

(End of definition for `\pnheading`.)

format option

```

143 \tl_new:N \l__postnotes_print_format_tl
144 \keys_define:nn { postnotes/setup }
145 {
146   format .tl_set:N = \l__postnotes_print_format_tl ,
147   format .initial:n = { \small } ,
148   format .value_required:n = true ,
149 }

```

listenv option

```

150 \tl_new:N \l__postnotes_print_env_tl
151 \bool_new:N \l__postnotes_print_as_list_bool
152 \keys_define:nn { postnotes/setup }
153 {
154   listenv .code:n =
155     {
156       \tl_if_eq:nnTF {#1} { none }
157         {
158           \bool_set_false:N \l__postnotes_print_as_list_bool
159           \tl_set:Nn \l__postnotes_post_printnote_tl { \par }

```

A sensible default just in case. It should not get to be used though.

```

160       \tl_set:Nn \l__postnotes_print_env_tl { itemize }
161     }
162     {
163       \bool_set_true:N \l__postnotes_print_as_list_bool
164       \tl_set:Nn \l__postnotes_print_env_tl {#1}
165     }
166   } ,
167   listenv .initial:n = { postnoteslist } ,
168   listenv .value_required:n = true ,

```

169 }
}

A couple of built-in list environments provided for convenience, and `postnoteslist` as default. The horizontal setup of the label in these lists is based on the description environment of the standard classes (see the *The L^AT_EX Companion*).

```
170 \NewDocumentEnvironment { postnoteslist } { }  
171 {  
172   \list { }  
173   {  
174     \setlength { \leftmargin } { Opt }  
175     \setlength { \labelwidth } { Opt }  
176     \setlength { \itemindent } { .5\parindent }  
177     \cs_set_eq:NN \makelabel \_postnotes_list_makelabel:n  
178     \setlength { \rightmargin } { Opt }  
179     \setlength { \listparindent } { \parindent }  
180     \setlength { \parsep } { \parskip }  
181     \setlength { \itemsep } { Opt }  
182     \setlength { \topsep } { .5\topsep }  
183     \setlength { \partopsep } { .5\partopsep }  
184   }  
185 }  
186 { \endlist }  
187 \NewDocumentEnvironment { postnoteslisthang } { }  
188 {  
189   \list { }  
190   {  
191     \setlength { \leftmargin } { 1em }  
192     \setlength { \labelwidth } { -\leftmargin }  
193     \setlength { \itemindent } { -2\leftmargin }  
194     \cs_set_eq:NN \makelabel \_postnotes_list_makelabel:n  
195     \setlength { \rightmargin } { Opt }  
196     \setlength { \listparindent } { \parindent }  
197     \setlength { \parsep } { \parskip }  
198     \setlength { \itemsep } { Opt }  
199     \setlength { \topsep } { .5\topsep }  
200     \setlength { \partopsep } { .5\partopsep }  
201   }  
202 }  
203 { \endlist }  
204 \cs_new:Npn \_postnotes_list_makelabel:n #1  
205 { \hspace { \labelsep } \normalfont ~ #1 }
```

makemark and maketextmark options

The arguments are: #1 is the mark, #2 and #3 are, respectively, the start and the end of the backlink.

```
206 \keys_define:nn { postnotes/setup }  
207 {  
208   makemark .cs_set:Np = \_postnotes_make_mark:nnn #1#2#3 ,  
209   makemark .value_required:n = true ,
```

From the default kernel definition of `\@makefnmark`.

```
210   makemark .initial:n =  
211     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,  
212   maketextmark .cs_set:Np = \_postnotes_make_text_mark:nnn #1#2#3 ,
```

```

213     maketextmark .value_required:n = true ,
214     maketextmark .initial:n = { #2 #1 . #3 } ,
215 }

```

pretextmark, posttextmark, postprintnote options

```

216 \tl_new:N \l__postnotes_pre_textmark_tl
217 \tl_new:N \l__postnotes_post_textmark_tl
218 \tl_new:N \l__postnotes_post_printnote_tl
219 \keys_define:nn { postnotes/setup }
220 {
221     pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
222     pretextmark .value_required:n = true ,
223     posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
224     posttextmark .value_required:n = true ,
225     postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
226     postprintnote .value_required:n = true ,
227 }

```

hyperref and backlink options

```

228 \bool_new:N \l__postnotes_hyperlink_bool
229 \bool_new:N \l__postnotes_hyperref_warn_bool
230 \bool_new:N \l__postnotes_backlink_bool
231 \keys_define:nn { postnotes/setup }
232 {
233     hyperref .choice: ,
234     hyperref / auto .code:n =
235     {
236         \bool_set_true:N \l__postnotes_hyperlink_bool
237         \bool_set_false:N \l__postnotes_hyperref_warn_bool
238     } ,
239     hyperref / true .code:n =
240     {
241         \bool_set_true:N \l__postnotes_hyperlink_bool
242         \bool_set_true:N \l__postnotes_hyperref_warn_bool
243     } ,
244     hyperref / false .code:n =
245     {
246         \bool_set_false:N \l__postnotes_hyperlink_bool
247         \bool_set_false:N \l__postnotes_hyperref_warn_bool
248     } ,
249     hyperref .initial:n = auto ,
250     hyperref .default:n = true ,
251     backlink .bool_set:N = \l__postnotes_backlink_bool ,
252     backlink .initial:n = true ,
253     backlink .default:n = true ,
254 }
255 \AddToHook { begindocument }
256 {
257     \IfPackageLoadedTF { hyperref }
258     { }
259     {
260         \bool_if:NT \l__postnotes_hyperref_warn_bool

```



```

261     { \msg_warning:nn { postnotes } { missing-hyperref } }
262     \bool_set_false:N \l__postnotes_hyperlink_bool
263   }
264   \keys_define:nn { postnotes/setup }
265   {
266     hyperref .code:n =
267     {
268       \msg_warning:nnn { postnotes }
269       { option-preamble-only } { hyperref }
270     } ,
271     backlink .code:n =
272     {
273       \msg_warning:nnn { postnotes }
274       { option-preamble-only } { backlink }
275     } ,
276   }
277 }
278 \msg_new:nnn { postnotes } { option-preamble-only }
279 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
280 \msg_new:nnn { postnotes } { missing-hyperref }
281 { Missing~'hyperref'~package.~Setting~'hyperref=false'. }

```

sort option

```

282 \bool_new:N \l__postnotes_sort_bool
283 \keys_define:nn { postnotes/setup }
284 {
285   sort .bool_set:N = \l__postnotes_sort_bool ,
286   sort .initial:n = true ,
287   sort .default:n = true ,
288 }

```

style option

```

289 \keys_define:nn { postnotes/setup }
290 {
291   style .choice: ,
292   style / endnotes .meta:n =
293   {
294     listenv = none ,
295     format =
296     {
297       \footnotesize
298       \setlength { \rightskip } { Opt }
299       \setlength { \leftskip } { Opt }
300       \setlength { \parindent } { 1.8em }
301     } ,
302     pretextmark = { \par } ,

```

endnotes uses a zero width box to get the desired alignment to the right, but that does not play well with the backlinks, so we have a little more work to do to get this right.

```

303     maketextmark =
304     {
305       \hbox_set:Nn \l_tmpa_box { \@textsuperscript { \normalfont ##1 } }
306       \skip_horizontal:n { - \box_wd:N \l_tmpa_box }
307       ##2 \box_use:N \l_tmpa_box ##3

```

```

308     } ,
309   } ,
310   style / pagenote .meta:n =
311   {
312     listenv = none ,
313     format = { } ,
314     pretextmark = { \par\noindent } ,
315     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
316     posttextmark = { ~ } ,
317   } ,
318 }

```

`\postnotesetup`

`\postnotesetup` Provide `\postnotesetup`.

```
\postnotesetup{options}
```

```

319 \NewDocumentCommand \postnotesetup { m }
320 { \keys_set:nn { postnotes/setup } {#1} }

```

(End of definition for `\postnotesetup`.)

4 `\postnote`

Different from the traditional `\footnotemark` / `\footnotetext` system, in the context of end notes, the functionality which corresponds to `\footnotetext` is simply to store the data to be typeset later. Hence, some of the problems that afflict footnotes do not apply to end notes. Namely, and as far as I can tell, they can be used in “inner horizontal mode” (`\mbox` etc.), and in math mode, and if the “text” will be typeset in the same page as the “mark” is of little concern.

However, the separation between “mark” and “text” is still useful in other contexts: floats and contexts where multiple typesetting passes are performed. David Carlisle and Ulrike Fischer shared some thoughts on the matter at the TeX.SX chat: <https://chat.stackexchange.com/transcript/message/60754383#60754383>.

The interesting questions here are: if they are replaceable in their roles in these contexts and how much would we lose by providing them. In analyzing this, we have to distinguish two situations: when `\footnotemark` is called with no argument (and thus steps the counter), and when it is called with the optional argument (and thus refrains from stepping the counter).

For floats, the problem they pose is that they may disturb the *ordering* of the notes. This particular issue can be solved by using `\footnotemark` without argument, and manually adjusting the counter on subsequent calls to `\footnotetext`. A good example of the technique is <https://tex.stackexchange.com/a/43694>. True, a user may wish to specify the mark explicitly, but doesn’t necessarily need to do it to solve the ordering issue.

Multiple typesetting passes of content are much harder. And they abound: the standard classes’ `\caption` typesets the caption once, if it is short, but twice if it is longer than a line; `amsmath`’s math environments perform a measuring pass before actually typesetting the equations; `amsmath`’s `\text` macro runs the contents through `\mathchoice` (which typesets the contents four times) when in math mode; `tabularx` and `tabularray` also perform measuring passes of their tables; so does `csquotes`’ blockquotes; and certainly

more that I'm unaware. A number of these places offer some one or another way to mitigate the issue: `amsmath`, `tabularx`, `csquotes` and (optionally) `tabularray` restore counter values after measuring steps; `amsmath` offers a boolean to indicate when it is a measuring pass; `csquotes` offers further handles. But the standard `\caption` offers none, and neither does `amsmath`'s `\text` macro. Well, the `pkgcaption` package can disable the multiple passes for `\caption` with the option `singlelinecheck`, but it is not reasonable to require it for our purposes, so we must assume the worst case.

Enrico Gregorio is categorical in stating that `\endnotemark` and `\endnotetext` are required for `enotez` to handle `\caption`, which apparently it didn't offer originally: "The package should implement `\endnotemark` and `\endnotetext` for this case. According to the documentation, the author deems them to not be needed: he's wrong." (<https://tex.stackexchange.com/a/314937>). See also <https://tex.stackexchange.com/a/43794> and <https://tex.stackexchange.com/a/358207>.

In this scenario, when there's no way around the multiple passes, `\footnotemark` can only handle the general case if used with an argument, precisely because it inhibits the stepping of the counter. Otherwise the counter is stepped multiple times, and we'd get the wrong number (and mark). In some circumstances, if we know the number of passes is deterministic, we might get away by adjusting the counter manually (`\caption` may be dealt with this way: if we know it to be two lines, we can decrement the counter before it and get correct results, even hyperlinked). But in cases which adjusting the counter is sufficient, end notes can be dealt with in the same way, and doesn't need the separation between "mark" and "text". So, what is distinctive of the kernel's footnote apparatus, which allows it as much flexibility as one would like, is receiving an arbitrary number as argument and not stepping the counter. And as far as `\footnotemark` and `\footnotetext` are concerned, the main point of the optional argument is really to "manually establish the relation" between the two of them. So, if *not stepping the counter* is what is needed to handle the general case, is it viable to do so? What would we lose in so doing?

When receiving an arbitrary number as argument, as the kernel functionality for footnotes and other endnotes packages do, this value is expected to be printed as such, hence it must correspond to the `postnote` counter (in our case). But this counter is in the hands of the user, and can be reset along the document, thus its uniqueness cannot be ensured. But not stepping `postnote` is perfectly viable, as it just aims at storing how the mark is to be typeset. However, not stepping the ID counter would complicate things considerably. Not doing so implies we'd lose the connection we have between the "mark" and the corresponding "text". We might add the "text" to the queue, but all the metadata would be lost, including the `hyperref` anchor, but really the set of data without which the kind of functionality offered would be nonviable, or severely hampered. Not stepping `postnote` but stepping the ID counter also is not sufficient, because we'd get a note in duplicity. We could naively think that a gap in the ID is not a problem, and just not add the duplicate to the queue. But how could we tell the difference between a legitimate and an illegitimate step of the ID counter?

I have not been able to devise a way to "reconnect" "text" and "mark" in the absence of the unique ID counter. The most promising idea was to have mandatory arguments to `\postnotemark` and `\postnotetext` receiving a *label* which we could use to identify their counterparts, but I was not able to go through with this, and the attempts all increased complexity considerably. It is not just a label/ref system, there's got to be a one-to-one correspondence between the sets, uniqueness has to be ensured on both sides, and there cannot be "lone" marks or texts (a bijection). Besides, this label based system of identification would have to live side-by-side with the one based on the counter.

So, even if we'd have unique IDs, we wouldn't know beforehand in what form it comes. Considering the ID is used to build the variable name in which we store the note's information, this would also complicate things.

Besides, there are ways to get things working with multiple passes without the “mark”/“text” partition. As mentioned, there are a number of cases which offer some kind of “handle” or way to identify the multiple passes. `csquotes` has a dedicated hook that can be used. `amsmath` sets the `measuring@` boolean (which `hyperref` also defines). So, not all cases are as tricky as `\caption` or `\text`, and even that can be decently dealt with without a separation between “mark” and “text”. Besides, in difficult cases, the package offers a `nomark` option to `\postnote` to place a note, but typeset no mark. Then we can typeset a mark with `\postnoteref` referring to a `\label` in the note of interest. This would result in a correct mark without duplicity, and in a correct link from there to the note's text at `\printpostnotes`. The drawback is that the placement of `\postnote` would be important, and results sensitive to it. All the metadata is collected at the point of `\postnote`, anchor included, not at the point of `\postnoteref`. So the consequences are a slightly off backlink, possibly imprecise metadata, etc. Considering `hyperref` itself shies away completely from linking `\footnotemark` with an argument, I'd say there's some gain.

The truth is there are some trade-offs, there's no “ideal” solution. Still, all in all, my judgment is that the unique ID counter is worth more than the inconveniences of an occasional `\postnote[nomark]` referenced with `\postnoteref`, and even that should not be needed much. So, for the time being, until something else shakes this balance, I won't be offering `\postnotemark` and `\postnotetext`.

For the `hyperref` support for cross-references in `\postnote`, I've moved back and forth quite a lot. One of the ideas I fancied was using `\refstepcounter` and let `hyperref` do its job. But, since I want to have control of the anchor/destination name on both “sides”, I'd have to set `\theHpostnote` locally before calling `\refstepcounter`, otherwise results might sensitive to user calls to `\counterwithin` (see <https://github.com/latex3/hyperref/issues/230>, thanks Ulrike Fischer). However, even if that worked well for the default case, we still had to setup things manually, in case of a manually supplied mark. All in all, I'm just calling `\stepcounter`, setting the relevant cross-reference variables once and setting the anchor manually.

`postnote` is the public, user facing, counter for `\postnote`. It determines how the note's mark gets to be typeset. It can be reset, set, and have its printed representation changed. Of course, whether those are meaningful is up to the user.

```
321 \newcounter { postnote }
```

```
\g__postnotes_note_id_int \g__postnotes_note_id_int is the internal, unique counter which provides the ID number
of each note. It ties “mark” and “text” together, is also the connection between each
\l_postnotes_note_id_tl note and its data, including the content, which is stored in a property list named according
to \__postnotes_data_name:n and the ID number. \l_postnotes_note_id_tl is a
\g__postnotes_queue_seq convenience variable storing the counter's value. \g__postnotes_queue_seq stores the
sequence of notes' IDs to be processed by the next call of \printpostnotes.
```

```
322 \int_new:N \g__postnotes_note_id_int
```

```
323 \tl_new:N \l_postnotes_note_id_tl
```

```
324 \tl_set:Nn \l_postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
```

```
325 \seq_new:N \g__postnotes_queue_seq
```

(End of definition for `\g__postnotes_note_id_int`, `\l_postnotes_note_id_tl`, and `\g__postnotes_queue_seq`. This function is documented on page ??.)

`\postnote` Provide `\postnote`.

```

\postnote [options] {note text}
326 \NewDocumentCommand \postnote { 0 { } +m }
327   { \__postnotes_note:nn {#1} {#2} }

```

(End of definition for `\postnote`.)

`__postnotes_note:nn` The internal version of `\postnote`. The `postnotes/note/begin` hook is meant to provide a place from where some additional setup for the note can be performed. This is being used for adding support for some features/packages, but can also be used, for example, to add an extra local property for `zref`.

```

\__postnotes_note:nn[options]{note content}
328 \NewHook { postnotes/note/begin }
329 \cs_new_protected:Npn \__postnotes_note:nn #1#2
330   {
331     \group_begin:
332     \keys_set:nn { postnotes/note } {#1}
333     \__postnotes_inhibit_note:F
334     {
335       \int_gincr:N \g__postnotes_note_id_int
336       \tl_if_empty:NT \l__postnotes_mark_tl
337         {
338           \stepcounter { postnote }
339           \tl_set:Nx \l__postnotes_mark_tl { \thepostnote }
340         }
341       \seq_gput_right:Nx \g__postnotes_queue_seq
342         { \l_postnotes_note_id_tl }
343       \UseHook { postnotes/note/begin }
344       \cs_set:Npn \@currentcounter { postnote }
345       \cs_set:Npx \@currentlabel { \p@postnote \l__postnotes_mark_tl }
346       \MakeLinkTarget* { postnote. \l_postnotes_note_id_tl .mark }
347       \__postnotes_set_mark_page_label:x { \l_postnotes_note_id_tl }
348       \__postnotes_set_user_labels:
349       \bool_if:NF \l__postnotes_nomark_bool
350         {
351           \__postnotes_typeset_mark:xV
352             { \l_postnotes_note_id_tl } \l__postnotes_mark_tl
353         }
354       \__postnotes_store:nn { \l_postnotes_note_id_tl } {#2}
355     }
356     \group_end:
357   }

```

(End of definition for `__postnotes_note:nn`.)

Options for `\postnote`.

```

358 \tl_new:N \l__postnotes_mark_tl

```

```

359 \bool_new:N \l__postnotes_nomark_bool
360 \fp_new:N \l__postnotes_sort_num_fp
361 \tl_new:N \l__postnotes_note_label_tl
362 \bool_new:N \l__postnotes_manual_sortnum_bool
363 \bool_new:N \l__postnotes_maybe_multi_bool
364 \keys_define:nn { postnotes/note }
365 {
366   markstr .tl_set:N = \l__postnotes_mark_tl ,
367   markstr .value_required:n = true ,
368   sortnum .code:n =
369   {
370     \fp_set:Nn \l__postnotes_sort_num_fp {#1}
371     \bool_set_true:N \l__postnotes_manual_sortnum_bool
372   } ,
373   sortnum .value_required:n = true ,
374   mark .meta:n =
375   {
376     markstr = {#1} ,
377     sortnum = {#1} ,
378   } ,
379   mark .value_required:n = true ,
380   nomark .bool_set:N = \l__postnotes_nomark_bool ,
381   nomark .default:n = true ,
382   label .tl_set:N = \l__postnotes_note_label_tl ,
383   label .value_required:n = true ,
384 }

```

`__postnotes_inhibit_note:TF` In contexts of multiple passes of content, it may be needed, or preferred, to inhibit the note altogether to avoid side effects and duplicity. This conditional, obviously, will always return the true branch unless something is done in the `postnotes/note/inhibit` hook. This hook is meant to handle support for packages or features which may justify note inhibition, and the code there should set `\l__postnotes_inhibit_note_bool`, `\l__postnotes_print_plain_mark_bool` and `\l__postnotes_print_plain_mark_stepcounter_bool` as appropriate to the case.

```

385 \bool_new:N \l__postnotes_inhibit_note_bool
386 \bool_new:N \l__postnotes_print_plain_mark_bool
387 \bool_new:N \l__postnotes_print_plain_mark_stepcounter_bool
388 \NewHook { postnotes/note/inhibit }
389 \prg_new_protected_conditional:Npnn \__postnotes_inhibit_note: { F }
390 {
391   \bool_set_false:N \l__postnotes_inhibit_note_bool
392   \bool_set_false:N \l__postnotes_print_plain_mark_bool
393   \bool_set_false:N \l__postnotes_print_plain_mark_stepcounter_bool
394   \UseHook { postnotes/note/inhibit }

```

Printing a plain mark here may be needed because, if we are inhibiting the note in a “measuring context” and omit it completely, the measuring being performed will be off by the size of the mark. So, to ensure the measuring can be done correctly, we place the mark. What to do with the counter itself, depends on the situation. In places that are known to restore the counter values after the measuring pass, we can let the counter be stepped. And, actually we should do so, for example, in a `tabularx` with multiple postnotes, if we don’t step the counter, all the measuring will be done with the number of the first note. Otherwise, we don’t actually step the counter but, to typeset correctly the

mark that would be printed if the counter had been stepped, we increment `\c@postnote` locally and grouped, and smuggle `\thepostnote` out of the group.

```

395     \bool_if:NT \l__postnotes_print_plain_mark_bool
396     {
397         \tl_if_empty:NT \l__postnotes_mark_tl
398         {
399             \bool_if:NTF \l__postnotes_print_plain_mark_stepcounter_bool
400             {
401                 \stepcounter { postnote }
402                 \tl_set:Nx \l__postnotes_mark_tl { \thepostnote }
403             }
404             {
405                 \group_begin:
406                 \int_incr:N \c@postnote
407                 \exp_args:NNNx
408                 \group_end:
409                 \tl_set:Nn \l__postnotes_mark_tl { \thepostnote }
410             }
411         }
412         \__postnotes_typeset_mark_wrapper:n
413         { \__postnotes_make_mark:nnn { \l__postnotes_mark_tl } { } { } }
414     }
415     \bool_if:NTF \l__postnotes_inhibit_note_bool
416     { \prg_return_true: }
417     { \prg_return_false: }
418 }

```

(End of definition for `__postnotes_inhibit_note:TF`.)

`__postnotes_typeset_mark:nn` Auxiliary functions for mark typesetting in `__postnotes_note:nn`. `__postnotes_typeset_mark_wrapper:n` is based on the definition of `\@footnotemark` in the kernel.

```

\__postnotes_typeset_mark:nn {<note id>} {<mark>}
\__postnotes_typeset_mark_wrapper:n {<mark>}

419 \cs_new_protected:Npn \__postnotes_typeset_mark:nn #1#2
420 {
421     \__postnotes_typeset_mark_wrapper:n
422     {
423         \bool_if:NTF \l__postnotes_hyperlink_bool
424         {
425             \__postnotes_make_mark:nnn {#2}
426             { \hyper@linkstart { link } { postnote. #1 .text } }
427             { \hyper@linkend }
428         }
429         { \__postnotes_make_mark:nnn {#2} { } { } }
430     }
431 }
432 \cs_generate_variant:Nn \__postnotes_typeset_mark:nn { xV }
433 \tl_new:N \l__postnotes_saved_spacefactor_tl
434 \cs_new_protected:Npn \__postnotes_typeset_mark_wrapper:n #1
435 {
436     \mode_leave_vertical:
437     \mode_if_horizontal:T

```

```

438     {
439       \tl_set:Nx \l__postnotes_saved_spacefactor_tl { \the\spacefactor }
440       \nobreak
441     }
442     #1
443     \mode_if_horizontal:T
444     { \spacefactor \l__postnotes_saved_spacefactor_tl }
445     \scan_stop:
446   }

```

(End of definition for `__postnotes_typeset_mark:nn` and `__postnotes_typeset_mark_wrapper:n`.)

`__postnotes_set_user_labels:nn` Auxiliary function for user label setting in `__postnotes_note:nn`. Supports the `label` and `zlabel` options of `\postnote`.

```

447 \cs_new_protected:Npn \__postnotes_set_user_labels:
448   {
449     \tl_if_empty:NF \l__postnotes_note_label_tl
450     { \exp_args:NV \label \l__postnotes_note_label_tl }
451     \tl_if_empty:NF \l__postnotes_note_zlabel_tl
452     { \exp_args:NV \zlabel \l__postnotes_note_zlabel_tl }
453   }

```

(End of definition for `__postnotes_set_user_labels:.`)

5 `\postnoteref`

`\postnoteref` Provide `\postnoteref`.

```

\postnoteref{*}{\label}
454 \NewDocumentCommand \postnoteref { s m }
455   { \__postnotes_note_ref:nn {#1} {#2} }

```

(End of definition for `\postnoteref`.)

`__postnotes_note_ref:nn` The internal version of `\postnoteref`.

```

\__postnotes_note_ref:nn {\star bool} {\label}
456 \cs_new_protected:Npn \__postnotes_note_ref:nn #1#2
457   {
458     \group_begin:
459     \__postnotes_typeset_mark_wrapper:n
460     {
461       \bool_lazy_and:nnTF
462       { ! #1 }
463       { \l__postnotes_hyperlink_bool }
464       {
465         \hyperref [#2]
466         { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
467       }
468       { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
469     }
470     \group_end:
471   }

```


(End of definition for `_postnotes_note_ref:nn`.)

6 `\postnotesection`

`\postnotesection` Provide `\postnotesection` and `\postnotesectionx`.
`\postnotesectionx`

```
\postnotesection[<options>]{<section content>}
\postnotesectionx[<options>]{<section content>}

472 \NewDocumentCommand \postnotesection { 0 { } +m }
473 { \_postnotes_section:nn {#1} {#2} }
474 \NewDocumentCommand \postnotesectionx { 0 { } +m }
475 {
476   % NOTE Command deprecated in 2022-12-27 for v0.2.0.
477   \msg_warning:nn { postnotes } { postnotesectionx-deprecated }
478   \postnotesection [ #1 , exp ] {#2}
479 }
480 \msg_new:nnn { postnotes } { postnotesectionx-deprecated }
481 {
482   '\iow_char:N\postnotesectionx'~is-deprecated~\msg_line_context:..~
483   Use~the~'exp'~option~of~'\iow_char:N\postnotesection'~instead.
484 }
```

(End of definition for `\postnotesection` and `\postnotesectionx`.)

`_postnotes_section:nn` The internal version of `\postnotesection`.

```
\_postnotes_section:nn {<options>} {<content>}

485 \int_new:N \g__postnotes_sectid_int
486 \cs_new_protected:Npn \_postnotes_section:nn #1#2
487 {
488   \group_begin:
489   \int_gincr:N \g__postnotes_sectid_int
490   \int_gincr:N \g__postnotes_note_id_int
491   \seq_gput_right:Nx \g__postnotes_queue_seq { \l_postnotes_note_id_tl }
492   \tl_gclear:N \g__postnotes_section_name_tl
493   \keys_set:nn { postnotes/section } {#1}
494   \bool_if:NTF \l__postnotes_section_exp_bool
495     { \_postnotes_store_section:nx { \l_postnotes_note_id_tl } {#2} }
496     { \_postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
497   \group_end:
498 }
```

(End of definition for `_postnotes_section:nn`.)

Options for `\postnotesection`. Actually, I would have preferred to use “label” for the `name` option, but I feared I might need it further down the road for the traditional meaning.

```
499 \tl_new:N \g__postnotes_section_name_tl
500 \bool_new:N \l__postnotes_section_exp_bool
501 \keys_define:nn { postnotes/section }
502 {
```

```

503     name .tl_gset:N = \g__postnotes_section_name_tl ,
504     name .value_required:n = true ,
505     exp .bool_set:N = \l__postnotes_section_exp_bool ,
506     exp .initial:n = false ,
507     exp .default:n = true ,
508 }

```

7 \printpostnotes

`\printpostnotes` Provide `\printpostnotes`.

```

\printpostnotes

509 \NewDocumentCommand \printpostnotes { }
510 { \__postnotes_print_notes: }

```

(End of definition for \printpostnotes.)

<code>\pnthechapter</code>	User facing variables, aimed at making available some of the notes' and sections' metadata
<code>\pnthesection</code>	for the user at specific contexts.
<code>\pnthechapternextnote</code>	511 \tl_new:N \pnthechapter
<code>\pnthesectionnextnote</code>	512 \tl_new:N \pnthesection
<code>\pnthepage</code>	513 \tl_new:N \pnthechapternextnote
	514 \tl_new:N \pnthesectionnextnote
	515 \tl_new:N \pnthepage

(End of definition for \pnthechapter and others.)

<code>\g__postnotes_print_postnotes_int</code>	Auxiliary variables for <code>__postnotes_print_notes:</code> .
<code>\l__postnotes_print_note_id_tl</code>	516 \int_new:N \g__postnotes_print_postnotes_int
<code>\l__postnotes_print_note_id_next_tl</code>	517 \tl_new:N \l__postnotes_print_note_id_tl
<code>\l__postnotes_print_counter_tl</code>	518 \tl_new:N \l__postnotes_print_note_id_next_tl
<code>\l__postnotes_print_mark_tl</code>	519 \tl_new:N \l__postnotes_print_counter_tl
<code>\l__postnotes_print_type_curr_tl</code>	520 \tl_new:N \l__postnotes_print_mark_tl
<code>\l__postnotes_print_type_next_tl</code>	521 \tl_new:N \l__postnotes_print_type_curr_tl
<code>\l__postnotes_print_type_prev_tl</code>	522 \tl_new:N \l__postnotes_print_type_next_tl
<code>\l__postnotes_print_content_tl</code>	523 \tl_new:N \l__postnotes_print_type_prev_tl
<code>\l__postnotes_clear_queue_seq</code>	524 \tl_new:N \l__postnotes_print_content_tl
	525 \seq_new:N \l__postnotes_clear_queue_seq

(End of definition for \g__postnotes_print_postnotes_int and others. This function is documented on page ??.)

`__postnotes_print_notes:` hooks. Both meant at providing points of entry for additional setup, specially to add support to packages and features which require it. The `postnotes/print/begin` hook is run early in `__postnotes_print_notes:` and only once per call, after the user options have been processed. The `postnotes/print/note/begin` hook is run once for each note, at the point where environment variables are being set or restored, before the typesetting of either the mark or the text, but within a group of its own of each note.

```

526 \NewHook { postnotes/print/begin }
527 \NewHook { postnotes/print/note/begin }

```

The `postnotetext` is a counter used to restore the original value of `postnote` at the time of printing, for the purposes of cross-referencing, it should be different from `postnote` if a note may occur inside `\printpostnotes`. The `postnotesection` is a counter which is stepped for every postnote section which gets to be actually typeset. It's aim is to provide a valid "enclosing counter" to `postnote` in the context of `\printpostnotes`. Since we don't know where `postnote` may have been reset along the document, in the general case, we can't rely on any other preexisting counter. This means that the particular value of `postnotesection` is of little practical meaning, it really is just meant to provide recognizable "bounds" for `postnote` along the printing of the notes. Indeed, it is initialized to a very high value (larger than the conceivable number of postnote sections in a document), so that "marks" and "texts" don't mix in the same reference list, which would occur if the enclosing counters of both belonged to the same range, and with somewhat arbitrary results, since we cannot ensure the step of the enclosing counter along the document matches `postnotesection`. This is actually a tricky problem from the cross-referencing standpoint: two different things, which should be of the same type, are reset along the document, but shouldn't really be mixed together. They are both L^AT_EX 2_ε counters, since they may be required externally. Their main intended use case is to support `zref-clever`, but in principle they can be of general use.

```

528 \newcounter { postnotetext }
529 \newcounter { postnotesection }
530 \setcounter { postnotesection } { 10000 }

```

`__postnotes_print_notes`: The internal version of `\printpostnotes`.

```

\__postnotes_print_notes:
531 \cs_new_protected:Npn \__postnotes_print_notes:
532 {
533   \group_begin:
534   \int_gincr:N \g__postnotes_print_postnotes_int
535   \seq_if_empty:NTF \g__postnotes_queue_seq
536     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
537   {
538     \pnheading
539     \UseHook { postnotes/print/begin }
540     \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
541     \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_queue_seq
542     \__postnotes_verify_multipass:N \g__postnotes_queue_seq
543     \bool_if:NT \l__postnotes_sort_bool
544       { \__postnotes_sort_queue:N \g__postnotes_queue_seq }
545     \bool_gset_true:N \g__postnotes_header_vars_next_bool
546     \__postnotes_get_headers_data:N \g__postnotes_queue_seq
547     \__postnotes_set_headers_vars_first:

```

Ensure the first note after a heading has paragraph indentation when `listenv` is `none`. `endnotes` uses a workaroundsish solution in `\noteheading`, setting a box and then skipping back a line. Enrico Gregorio is correct, though, in criticizing it at https://tex.stackexchange.com/q/575905#comment1450213_575915, and suggests the use of `\@afterindenttrue`, which is what `indentfirst` does (we do the same, just locally).

```

548     \bool_if:NF \l__postnotes_print_as_list_bool
549     {

```

```

550         \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
551         \@afterindenttrue
552     }
553 \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_queue_seq }
554 {
555     \seq_gpop_left:NN \g__postnotes_queue_seq
556     \l_postnotes_print_note_id_tl
557     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
558     { type } \l_postnotes_print_type_curr_tl
559     \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
560     { % type_curr = 'section'
561         \seq_if_empty:NTF \g__postnotes_queue_seq
562         {
563             \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
564             \tl_set:Nn \l__postnotes_print_type_next_tl { close }
565         }
566         {
567             \seq_get_left:NN \g__postnotes_queue_seq
568             \l__postnotes_print_note_id_next_tl
569             \__postnotes_prop_get:nnN
570             { \l__postnotes_print_note_id_next_tl }
571             { type } \l__postnotes_print_type_next_tl
572         }
573     }

```

We only process the entry if type_next is note: here are skipped empty sections.

```

573     \tl_if_eq:NnTF \l__postnotes_print_type_next_tl { note }
574     {
575         \stepcounter { postnotesection }
576         \group_begin:
577         \__postnotes_prop_get:nnN
578         { \l_postnotes_print_note_id_tl }
579         { thechapter } \pnthechapter
580         \__postnotes_prop_get:nnN
581         { \l_postnotes_print_note_id_tl }
582         { thesection } \pnthesection
583         \__postnotes_prop_get:nnN
584         { \l__postnotes_print_note_id_next_tl }
585         { thechapter } \pnthechapternextnote
586         \__postnotes_prop_get:nnN
587         { \l__postnotes_print_note_id_next_tl }
588         { thesection } \pnthesectionnextnote
589         \__postnotes_prop_get:nnN
590         { \l_postnotes_print_note_id_tl }
591         { content } \l__postnotes_print_content_tl
592         \l__postnotes_print_content_tl
593         \group_end:

```

Set type_prev for the next iteration.

```

594         \tl_set:NV \l__postnotes_print_type_prev_tl
595         \l__postnotes_print_type_curr_tl
596     }
597 }
598 { % type_curr = 'note'
599     \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }
600     {

```

```

601         \bool_if:NTF \l__postnotes_print_as_list_bool
602         { \exp_args:Nx \begin { \l__postnotes_print_env_tl } }
603         { \group_begin: }
604         \l__postnotes_print_format_tl
605     }
606     \group_begin:
607     \UseHook { postnotes/print/note/begin }
608     \__postnotes_get_pageref:Nx \pnthepage
609     { mark@ \l_postnotes_print_note_id_tl }
610     \__postnotes_prop_get:nnN
611     { \l_postnotes_print_note_id_tl }
612     { mark } \l__postnotes_print_mark_tl
613     \__postnotes_prop_get:nnN
614     { \l_postnotes_print_note_id_tl }
615     { counter } \l__postnotes_print_counter_tl
616     \__postnotes_prop_get:nnN
617     { \l_postnotes_print_note_id_tl }
618     { content } \l__postnotes_print_content_tl
619     \cs_set:Npn \@currentcounter { postnotetext }
620     \int_set:Nn \c@postnotetext
621     { \l__postnotes_print_counter_tl }
622     \cs_set:Npx \@currentlabel
623     { \p@postnote \l__postnotes_print_mark_tl }
624     \__postnotes_text_mark_wrapper:n
625     {
626         \MakeLinkTarget*
627         { postnote. \l_postnotes_print_note_id_tl .text }
628         \__postnotes_set_text_page_label:x
629         { \l_postnotes_print_note_id_tl }
630         \__postnotes_typeset_text_mark:eV
631         { \l_postnotes_print_note_id_tl }
632         \l__postnotes_print_mark_tl
633     }
634     \l__postnotes_print_content_tl
635     \l__postnotes_post_printnote_tl
636     \group_end:

```

For notes, query for next note’s type after the current note was typeset, to handle possible nesting. Even if nesting is not a feature, this should avoid hard crashes related to “lonely \item” or “extra \endgroup” errors, in case it occurs.

```

637     \seq_if_empty:NTF \g__postnotes_queue_seq
638     {
639         \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
640         \tl_set:Nn \l__postnotes_print_type_next_tl { close }
641     }
642     {
643         \seq_get_left:NN \g__postnotes_queue_seq
644         \l__postnotes_print_note_id_next_tl
645         \__postnotes_prop_get:nnN
646         { \l__postnotes_print_note_id_next_tl }
647         { type } \l__postnotes_print_type_next_tl
648     }
649     \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
650     {

```

```

651         \bool_if:NTF \l__postnotes_print_as_list_bool
652         { \exp_args:Nx \end { \l__postnotes_print_env_tl } }
653         { \group_end: }

```

Ensure `\par` at the end of `\printpostnotes` (see <https://github.com/u-fischer/tagpdf/issues/68#issuecomment-1587343876>, thanks Ulrike Fischer).

```

654         \par
655     }

```

Set `type_prev` for the next iteration.

```

656         \tl_set:NV \l__postnotes_print_type_prev_tl
657         \l__postnotes_print_type_curr_tl
658     }
659 }
660 \AddToHookNext { shipout/after }
661 { \bool_gset_false:N \g__postnotes_header_vars_next_bool }

```

We won't use the variables anymore, clear them to reduce memory usage. Given how we populated `\l__postnotes_clear_queue_seq`, this won't catch nested notes. But it's not worth to conditionally add new items along the way (testing it every iteration) for this. Again, not a feature.

```

662     \seq_map_inline:Nn \l__postnotes_clear_queue_seq
663     { \__postnotes_prop_gclear:n { ##1 } }
664 }
665 \group_end:
666 }

```

(End of definition for `__postnotes_print_notes:.`)

```

667 \msg_new:nnn { postnotes } { empty-printpostnotes }
668 { Empty-'\iow_char:N\printpostnotes'~\msg_line_context:. }

```

`__postnotes_typeset_text_mark:nn`
`__postnotes_text_mark_wrapper:n` Auxiliary functions for mark typesetting in `__postnotes_print_notes:.`

```

        \__postnotes_typeset_text_mark:nn {<note id>} {<mark>}
        \__postnotes_text_mark_wrapper:n {<mark>}

669 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
670 {
671     \bool_lazy_and:nnTF
672     { \l__postnotes_hyperlink_bool }
673     { \l__postnotes_backlink_bool }
674     {
675         \__postnotes_make_text_mark:nnn {#2}
676         { \hyper@linkstart { link } { postnote. #1 .mark } }
677         { \hyper@linkend }
678     }
679     { \__postnotes_make_text_mark:nnn {#2} { } { } }
680 }
681 \cs_generate_variant:Nn \__postnotes_typeset_text_mark:nn { eV }
682 \cs_new_protected:Npn \__postnotes_text_mark_wrapper:n #1
683 {
684     \bool_if:NTF \l__postnotes_print_as_list_bool
685     {
686         \item [ \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl ]

```

Leave vertical mode to avoid “perhaps a missing `\item`” error for empty notes.

```

687     \mode_leave_vertical:
688     }
689     { \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl }
690   }

```

(End of definition for `__postnotes_typeset_text_mark:n` and `__postnotes_text_mark_wrapper:n`.)

Print auxiliary

`__postnotes_verify_multipass:N` provides a general procedure for handling cases of multiple passes of content. Ideally, the job should be done at `__postnotes_inhibit_note:F`, if at all possible. But, failing that, we can rely on the fact that `\postnotes` of measuring/trial passes don’t end up being output and hence don’t generate labels in the `.aux` file. This is the equivalent for `postnotes` to the effect of write restrictions for the packages based on external files, which is how they actually handle these cases. However, despite this being a general test, and a reasonable one, I’d like to restrain its use to the minimum possible. First, using this criterion across the board would result in large swings on the content of `\printpostnotes` and spurious warnings in an initial compilation since the labels are not available on the first run. Second, I’d prefer not to interfere with the queue, unless we really need to. Hence, we only apply this check for “eligible” items. For signaling this eligibility, the note must have been stored with the `\l__postnotes_maybe_multi_bool` set to `true`, which is then saved in the `multibool` property. One implication of this procedure is that, if there are any new notes marked as `multibool`, three rounds of compilation will be needed, since the labels of the printed notes will be written only on the second run and the document will thus require a third one to stabilize.

```

\__postnotes_verify_multipass:N     \__postnotes_verify_multipass:N (\g__postnotes_queue_seq)
691 \cs_new_protected:Npn \__postnotes_verify_multipass:N #1
692   {
693     \group_begin:
694     \seq_clear:N \l_tmpa_seq
695     \seq_map_inline:Nn #1
696     {
697       \__postnotes_prop_get:nnN {##1} { multibool } \l_tmpa_tl
698       \tl_if_eq:NnTF \l_tmpa_tl { true }
699       {
700         \cs_if_exist:cT
701         { \c__postnotes_ref_prefix_tl @ mark@ ##1 }
702         { \seq_put_right:Nn \l_tmpa_seq {##1} }
703       }
704       { \seq_put_right:Nn \l_tmpa_seq {##1} }
705     }
706     \seq_gset_eq:NN #1 \l_tmpa_seq
707     \group_end:
708   }

```

(End of definition for `__postnotes_verify_multipass:N`.)

`__postnotes_sort_queue:N` Sorting function for `__postnotes_print_notes:.`

```

709 \__postnotes_sort_queue:N (\g__postnotes_queue_seq)
710 {
711   \group_begin:
712   \seq_gsort:Nn #1
713   {
714     \__postnotes_prop_get:nnN {##1} { pnsectid } \l_tmpa_tl
715     \__postnotes_prop_get:nnN {##2} { pnsectid } \l_tmpb_tl
716     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
717     {
718       \__postnotes_prop_get:nnN {##1} { type } \l_tmpa_tl
719       \__postnotes_prop_get:nnN {##2} { type } \l_tmpb_tl
720       \bool_lazy_and:nnTF
721       { \str_if_eq_p:Vn \l_tmpa_tl { note } }
722       { \str_if_eq_p:Vn \l_tmpb_tl { note } }
723       {
724         \__postnotes_prop_get:nnN {##1} { sortnum } \l_tmpa_tl
725         \__postnotes_prop_get:nnN {##2} { sortnum } \l_tmpb_tl
726         \fp_compare:nNnTF { \l_tmpa_tl } > { \l_tmpb_tl }
727         { \sort_return_swapped: }
728         { \sort_return_same: }
729       }
730       { \sort_return_same: }
731     }
732     { \sort_return_same: }
733   }
734   \group_end:
735 }

```

(End of definition for __postnotes_sort_queue:N.)

8 Headers

The headers infrastructure of `postnotes` is comprised of three basic parts:

1. For each `\postnote`, labels are set storing the `page` where the note occurs. Each note actually generates a pair of such labels, once when `\postnote` is called (with the mark), and another where the note is printed (in `\printpostnotes`). The former ones store `\thepage`, since we want the printed representation of it for typesetting purposes, the latter ones store the value of the `page` counter, since we don't need to typeset it, but do need to perform algebraic operations with it. These labels are set by `__postnotes_set_mark_page_label:n`, `__postnotes_set_text_page_label:n`, and `__postnotes_set_print_page_label:n` at the appropriate places. The set of these labels provides a mapping from each note's "mark" and "text" to the page where it occurs.
2. This information set is processed by `__postnotes_get_headers_data:N` at the beginning of `__postnotes_print_notes:` to identify the first and last note of each page in `\printpostnotes`, and to generate a mapping from these first and last notes on each page to the pages where their corresponding marks occur. We also take the opportunity to enrich this mapping with other metadata of each note. So we get also

mappings from the first and last note on each page to `\thechapter`, `\thesection`, and the name of the section in which they occur. These mappings are stored in property lists `\g__postnotes_header_⟨info⟩_first_prop` and `\g__postnotes_header_⟨info⟩_last_prop` where the key is the page in `\printpostnotes` where their note’s content is typeset (or rather where it starts to be typeset, it is the page where the text’s mark is printed).

3. Based on these mappings, along the span of notes section we run `__postnotes_set_headers_vars_next`: at each `shipout/before` hook to set user facing variables for the *next* page, which will be available when their heading gets typeset. Given that at `shipout` we can rely on a correct value of the page counter, we use it as key to query the property lists generated in the previous step. These user facing variables are called `\pnhd⟨info⟩first` and `\pnhd⟨info⟩last`. Since we cannot rely on the shipout hook for the first page of `\printpostnotes`, `__postnotes_set_headers_vars_first`: is run at its beginning to ensure correct values are in place on the first page of the notes section.

These `\pnhd⟨info⟩first` and `\pnhd⟨info⟩last` variables can then be used to build simple functions which can be passed to mark commands to achieve rich contextual running headers.

<code>\pnhdpagefirst</code>	User facing variables, aimed at making available header data for the user. Setting these
<code>\pnhdpagelast</code>	variables with correct values at the moment the header gets typeset is <i>the</i> objective of
<code>\pnhdchapfirst</code>	the whole headers infrastructure of the package.
<code>\pnhdchaplast</code>	
<code>\pnhdsectfirst</code>	736 <code>\tl_new:N \pnhdpagefirst</code>
<code>\pnhdsectlast</code>	737 <code>\tl_new:N \pnhdpagelast</code>
<code>\pnhdnamefirst</code>	738 <code>\tl_new:N \pnhdchapfirst</code>
<code>\pnhdnamelast</code>	739 <code>\tl_new:N \pnhdchaplast</code>
	740 <code>\tl_new:N \pnhdsectfirst</code>
	741 <code>\tl_new:N \pnhdsectlast</code>
	742 <code>\tl_new:N \pnhdnamefirst</code>
	743 <code>\tl_new:N \pnhdnamelast</code>

(End of definition for `\pnhdpagefirst` and others.)

<code>\g__postnotes_header_page_first_prop</code>	Auxiliary variables for the headers infrastructure.
<code>\g__postnotes_header_page_last_prop</code>	744 <code>\prop_new:N \g__postnotes_header_page_first_prop</code>
<code>\g__postnotes_header_chap_first_prop</code>	745 <code>\prop_new:N \g__postnotes_header_page_last_prop</code>
<code>\g__postnotes_header_chap_last_prop</code>	746 <code>\prop_new:N \g__postnotes_header_chap_first_prop</code>
<code>\g__postnotes_header_sect_first_prop</code>	747 <code>\prop_new:N \g__postnotes_header_chap_last_prop</code>
<code>\g__postnotes_header_sect_last_prop</code>	748 <code>\prop_new:N \g__postnotes_header_sect_first_prop</code>
<code>\g__postnotes_header_name_first_prop</code>	749 <code>\prop_new:N \g__postnotes_header_sect_last_prop</code>
<code>\g__postnotes_header_name_last_prop</code>	750 <code>\prop_new:N \g__postnotes_header_name_first_prop</code>
<code>\g__postnotes_header_prev_last_page_tl</code>	751 <code>\prop_new:N \g__postnotes_header_name_last_prop</code>
<code>\g__postnotes_header_prev_last_chap_tl</code>	752 <code>\tl_new:N \g__postnotes_header_prev_last_page_tl</code>
<code>\g__postnotes_header_prev_last_sect_tl</code>	753 <code>\tl_new:N \g__postnotes_header_prev_last_chap_tl</code>
<code>\g__postnotes_header_prev_last_name_tl</code>	754 <code>\tl_new:N \g__postnotes_header_prev_last_sect_tl</code>
<code>\l__postnotes_prev_text_page_tl</code>	755 <code>\tl_new:N \g__postnotes_header_prev_last_name_tl</code>
<code>\l__postnotes_curr_text_page_tl</code>	756 <code>\tl_new:N \l__postnotes_prev_text_page_tl</code>
<code>\l__postnotes_prev_mark_page_tl</code>	757 <code>\tl_new:N \l__postnotes_curr_text_page_tl</code>
<code>\l__postnotes_prev_mark_chap_tl</code>	758 <code>\tl_new:N \l__postnotes_prev_mark_page_tl</code>
<code>\l__postnotes_prev_mark_sect_tl</code>	759 <code>\tl_new:N \l__postnotes_prev_mark_chap_tl</code>
<code>\l__postnotes_prev_mark_name_tl</code>	

```

760 \tl_new:N \l__postnotes_prev_mark_sect_tl
761 \tl_new:N \l__postnotes_prev_mark_name_tl

(End of definition for \g__postnotes_header_page_first_prop and others.)

```

_postnotes_get_headers_data:N Process header data for _postnotes_set_headers_vars:n.

```

\_postnotes_get_headers_data:N (\g__postnotes_queue_seq)

762 \cs_new_protected:Npn \_postnotes_get_headers_data:N #1
763 {
764   \group_begin:
765   \tl_gclear:N \pnhdpagefirst
766   \tl_gclear:N \pnhdpagelast
767   \tl_gclear:N \pnhdchapfirst
768   \tl_gclear:N \pnhdchapl原因
769   \tl_gclear:N \pnhdsectfirst
770   \tl_gclear:N \pnhdsectlast
771   \tl_gclear:N \pnhdnamefirst
772   \tl_gclear:N \pnhdnamelast
773   \prop_gclear:N \g__postnotes_header_page_first_prop
774   \prop_gclear:N \g__postnotes_header_page_last_prop
775   \prop_gclear:N \g__postnotes_header_chap_first_prop
776   \prop_gclear:N \g__postnotes_header_chap_last_prop
777   \prop_gclear:N \g__postnotes_header_sect_first_prop
778   \prop_gclear:N \g__postnotes_header_sect_last_prop
779   \prop_gclear:N \g__postnotes_header_name_first_prop
780   \prop_gclear:N \g__postnotes_header_name_last_prop
781   \tl_gclear:N \g__postnotes_header_prev_last_page_tl
782   \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
783   \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
784   \tl_gclear:N \g__postnotes_header_prev_last_name_tl
785   \tl_clear:N \l__postnotes_prev_text_page_tl
786   \tl_clear:N \l__postnotes_curr_text_page_tl
787   \tl_clear:N \l__postnotes_prev_mark_page_tl
788   \tl_clear:N \l__postnotes_prev_mark_chap_tl
789   \tl_clear:N \l__postnotes_prev_mark_sect_tl
790   \tl_clear:N \l__postnotes_prev_mark_name_tl
791   \seq_map_inline:Nn #1
792   {
793     \exp_args:Nx \tl_if_eq:nnT
794     { \_postnotes_prop_item:nn {##1} { type } }
795     { note }
796     {
797       \_postnotes_get_pageref:Nn
798       \l__postnotes_curr_text_page_tl { text@ ##1 }
799       \tl_if_empty:NF \l__postnotes_curr_text_page_tl
800       {
801         \tl_if_eq:NNTF
802         \l__postnotes_prev_text_page_tl
803         \l__postnotes_curr_text_page_tl
804         {

```

We are on the same page as the previous note, just update the prev_mark data.

```

805         \_postnotes_get_pageref:Nn
806         \l__postnotes_prev_mark_page_tl { mark@ ##1 }
807     \_postnotes_prop_get:nnN {##1} { thechapter }
808         \l__postnotes_prev_mark_chap_tl
809     \_postnotes_prop_get:nnN {##1} { thesection }
810         \l__postnotes_prev_mark_sect_tl
811     \_postnotes_prop_get:nnN {##1} { pnsectname }
812         \l__postnotes_prev_mark_name_tl
813     }
814 {

```

We are on the transition between two pages, current ID is the first note of the new page (or on the very first note of \printpostnotes, given \l__postnotes_prev_text_page_tl is initialized to empty).

Set ‘last’ values for previous page, based on the last valid prev_mark stored ones. There is no previous page to the first one of \printpostnotes, so we don’t set ‘last’ values for it (conditioning on \l__postnotes_prev_text_page_tl being empty, which only occurs on the first note).

```

815     \tl_if_empty:NF \l__postnotes_prev_text_page_tl
816     {
817         \prop_gput:Nxx \g__postnotes_header_page_last_prop
818         { \l__postnotes_prev_text_page_tl }
819         { \l__postnotes_prev_mark_page_tl }
820     \prop_gput:Nxx \g__postnotes_header_chap_last_prop
821     { \l__postnotes_prev_text_page_tl }
822     { \l__postnotes_prev_mark_chap_tl }
823     \prop_gput:Nxx \g__postnotes_header_sect_last_prop
824     { \l__postnotes_prev_text_page_tl }
825     { \l__postnotes_prev_mark_sect_tl }
826     \prop_gput:Nxx \g__postnotes_header_name_last_prop
827     { \l__postnotes_prev_text_page_tl }
828     { \l__postnotes_prev_mark_name_tl }
829     }

```

Set ‘first’ values for current page, based on the current note ID.

```

830     \prop_gput:Nxx \g__postnotes_header_page_first_prop
831     { \l__postnotes_curr_text_page_tl }
832     { \_postnotes_extract_pageref:n { mark@ ##1 } }
833     \prop_gput:Nxx \g__postnotes_header_chap_first_prop
834     { \l__postnotes_curr_text_page_tl }
835     { \_postnotes_prop_item:nn {##1} { thechapter } }
836     \prop_gput:Nxx \g__postnotes_header_sect_first_prop
837     { \l__postnotes_curr_text_page_tl }
838     { \_postnotes_prop_item:nn {##1} { thesection } }
839     \prop_gput:Nxx \g__postnotes_header_name_first_prop
840     { \l__postnotes_curr_text_page_tl }
841     { \_postnotes_prop_item:nn {##1} { pnsectname } }

```

Store prev_mark data for the first note on the page.

```

842     \_postnotes_get_pageref:Nn
843     \l__postnotes_prev_mark_page_tl { mark@ ##1 }
844     \_postnotes_prop_get:nnN {##1} { thechapter }
845     \l__postnotes_prev_mark_chap_tl
846     \_postnotes_prop_get:nnN {##1} { thesection }
847     \l__postnotes_prev_mark_sect_tl

```

```

848         \l__postnotes_prop_get:nnN {##1} { pnsectname }
849         \l__postnotes_prev_mark_name_tl
      Set \l__postnotes_prev_text_page_tl for the next page (\l__postnotes_curr_
text_page_tl is never empty at this point, since we conditioned to it).
850         \tl_set:NV \l__postnotes_prev_text_page_tl
851         \l__postnotes_curr_text_page_tl
852     }
853 }
854 }
855 }

```

We can't catch the transition from the last page of `\printpostnotes` to the following one through the mapping above, but the `prev_mark` values of the last note in the loop are the ones we want, so we set 'last' values for the last page based on them.

```

856 \tl_if_empty:NF \l__postnotes_prev_text_page_tl
857 {
858     \prop_gput:Nxx \g__postnotes_header_page_last_prop
859     { \l__postnotes_prev_text_page_tl }
860     { \l__postnotes_prev_mark_page_tl }
861     \prop_gput:Nxx \g__postnotes_header_chap_last_prop
862     { \l__postnotes_prev_text_page_tl }
863     { \l__postnotes_prev_mark_chap_tl }
864     \prop_gput:Nxx \g__postnotes_header_sect_last_prop
865     { \l__postnotes_prev_text_page_tl }
866     { \l__postnotes_prev_mark_sect_tl }
867     \prop_gput:Nxx \g__postnotes_header_name_last_prop
868     { \l__postnotes_prev_text_page_tl }
869     { \l__postnotes_prev_mark_name_tl }
870 }
871 \group_end:
872 }

```

(End of definition for `__postnotes_get_headers_data:N`.)

The sequence of pages processed in `__postnotes_get_headers_data:N` is not ensured to be continuous, since not every page of `\printpostnotes` starts a note. There may be notes that fill whole pages, or the last page of the notes may end with a note that started on the penultimate page. We must handle this case at `__postnotes_set_headers_vars:n`. For every page for which there is information provided by `__postnotes_get_headers_data:N` we store a `header_prev_last` (the last value of the previous header) for each of the variables of interest. If the next page is skipped in the sequence (no notes starting on it), we can use these stored values to set both 'first' and 'last' variables based on them for that page.

`__postnotes_set_headers_vars:n` Set user facing variables based on data generated by `__postnotes_get_headers_data:N`.

```

      \__postnotes_set_headers_vars:n {(page number)}

873 \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
874 {
875     \group_begin:
876     \prop_get:NnNTF \g__postnotes_header_page_first_prop
877     {#1} \l_tmpa_tl

```

```

878     { \tl_gset:NV \pnhdpagefirst \l_tmpa_tl }
879     { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
880 \prop_get:NnNTF \g__postnotes_header_page_last_prop
881   {#1} \l_tmpa_tl
882   {
883     \tl_gset:NV \pnhdpagelast \l_tmpa_tl
884     \tl_gset:NV \g__postnotes_header_prev_last_page_tl \l_tmpa_tl
885   }
886   { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
887 \prop_get:NnNTF \g__postnotes_header_chap_first_prop
888   {#1} \l_tmpa_tl
889   { \tl_gset:NV \pnhdchapfirst \l_tmpa_tl }
890   { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
891 \prop_get:NnNTF \g__postnotes_header_chap_last_prop
892   {#1} \l_tmpa_tl
893   {
894     \tl_gset:NV \pnhdchaplast \l_tmpa_tl
895     \tl_gset:NV \g__postnotes_header_prev_last_chap_tl \l_tmpa_tl
896   }
897   { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
898 \prop_get:NnNTF \g__postnotes_header_sect_first_prop
899   {#1} \l_tmpa_tl
900   { \tl_gset:NV \pnhdsectfirst \l_tmpa_tl }
901   { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
902 \prop_get:NnNTF \g__postnotes_header_sect_last_prop
903   {#1} \l_tmpa_tl
904   {
905     \tl_gset:NV \pnhdsectlast \l_tmpa_tl
906     \tl_gset:NV \g__postnotes_header_prev_last_sect_tl \l_tmpa_tl
907   }
908   { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
909 \prop_get:NnNTF \g__postnotes_header_name_first_prop
910   {#1} \l_tmpa_tl
911   { \tl_gset:NV \pnhdnamefirst \l_tmpa_tl }
912   { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
913 \prop_get:NnNTF \g__postnotes_header_name_last_prop
914   {#1} \l_tmpa_tl
915   {
916     \tl_gset:NV \pnhdnamelast \l_tmpa_tl
917     \tl_gset:NV \g__postnotes_header_prev_last_name_tl \l_tmpa_tl
918   }
919   { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
920 \group_end:
921 }
922 \cs_generate_variant:Nn \__postnotes_set_headers_vars:n { x }

```

(End of definition for __postnotes_set_headers_vars:n.)

`__postnotes_set_headers_vars_next:` The functions that actually call `__postnotes_set_headers_vars:n` at the appropriate contexts with appropriate page values. Though we set `__postnotes_set_headers_vars_next:` to run at every shipout/before hook of the document, it is made no-op by `\g__postnotes_header_vars_next_bool` which only has a true value inside `\printpostnotes`. `__postnotes_set_headers_vars_first:` must set a label and retrieve its value to be able to have a reliable value of its own page.

```

923 \AddToHook { shipout/before } [ postnotes/header ]
924   { \_postnotes_set_headers_vars_next: }
925 \bool_new:N \g__postnotes_header_vars_next_bool
926 \cs_new_protected:Npn \_postnotes_set_headers_vars_next:
927   {
928     \bool_if:NT \g__postnotes_header_vars_next_bool
929       { \_postnotes_set_headers_vars:x { \int_eval:n { \c@page + 1 } } }
930   }
931 \cs_new_protected:Npn \_postnotes_set_headers_vars_first:
932   {
933     \_postnotes_set_print_page_label:x
934     { \int_use:N \g__postnotes_print_postnotes_int }
935     \_postnotes_set_headers_vars:x
936     {
937       \_postnotes_extract_pageref:e
938       { print@ \int_use:N \g__postnotes_print_postnotes_int }
939     }
940   }

```

(End of definition for _postnotes_set_headers_vars_next: and _postnotes_set_headers_vars_first:.)

`\pnheaderdefault` A basic header function to be used as default in the `heading` option. It produces a header in the form “Notes to pages N–M”, with a text which can be localized (see Section 10).

`\pnheaderdefault`

```

941 \NewDocumentCommand \pnheaderdefault {}
942   {
943     \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
944       { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
945       { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
946   }

```

(End of definition for \pnheaderdefault.)

9 Compatibility

A dedicated temp variable for restoring data.

```

947 \tl_new:N \l__postnotes_restore_tmp_tl

```

`\caption`

For `\caption`’s possible two passes. This catches more than just captions, of course, but is not overkill.

From the user’s perspective, one-line captions will just work. For two-line captions, there are two alternatives: i) decrement the counter by 1 `\addtocounter{postnote}{-1}` before the caption, then call `\postnote` inside the caption; or ii) right before the caption, call `\postnote[nomark]{\label{mynote}...}`, then use `\postnoteref{mynote}` inside the caption.

```

948 \AddToHook { postnotes/note/begin } [ postnotes ]
949   {

```

```

950 \cs_if_exist:NT \@capttype
951   { \bool_set_true:N \l__postnotes_maybe_multi_bool }
952 }

```

biblatex

Thanks Moritz Wemheuer: https://tex.stackexchange.com/q/597359#comment1594585_597389.

```

953 \AddToHook { package/biblatex/after }
954   {

```

Let biblatex know we are in a “notes” context. See <https://tex.stackexchange.com/a/304464>, including comments.

```

955   \AddToHook { postnotes/print/begin } [ postnotes ]
956   { \toggletrue { blx@footnote } }

```

Make biblatex’s `\mkbibendnote` use `\postnote`. This is very likely desired in most cases, but may occasionally not be, so we add it to an individually labeled hook, which can be disabled with `\RemoveFromHook{begindocument/before}[postnotes/mkbibendnote]` in the preamble.

```

957   \AddToHook { begindocument/before } [ postnotes/mkbibendnote ]
958   {
959     \cs_set:Npn \blx@theendnote { \postnote }
960     \cs_set:Npn \blx@theendnotetext
961       { \blx@err@endnote \footnotetext }
962   }
963 }
964 <*gobble>

```

I had made an initial experimental attempt to support biblatex’s `refsegments`, `refcontexts` and `refsections`. However, this attempt was rash. Even if I could get many example files to work for `refsegments` and `refcontexts`, I could not do so for `refsections`. More importantly, with this partial implementation, I could also generate documents which confused biblatex more than it helped. Things I couldn’t understand well, or fix. All in all, I don’t think this partial implementation is tenable, and I could not take it further. Hence, `postnotes` support for this feature set of biblatex will depend, as it should, on proper upstream support for “saving” and “restoring” citation “context” information.

I have made a feature request at biblatex for this (<https://github.com/plk/biblatex/issues/1226>), which was (understandably) classified as “long term, no promises”.

The attempt was the following (currently “gobbled” from the package):

```

965 \AddToHook { package/biblatex/after }
966   {

```

Store biblatex variables for each note.

```

967   \AddToHook { postnotes/note/store } [ postnotes ]
968   {
969     \prop_gput:cnx { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
970     { biblatex@refsection } { \int_use:N \c@refsection }
971     \prop_gput:cnx { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
972     { biblatex@refsegment } { \int_use:N \c@refsegment }

```

```

973     \prop_gput:cnx { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
974     { biblatex@refcontextbool }
975     { \iftoggle { blx@refcontext } { true } { false } }
976     \prop_gput:cnV { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
977     { biblatex@refcontext } \blx@refcontext@context
978   }

```

biblatex setup, once for \printpostnotes call.

```

979   \AddToHook { postnotes/print/begin } [ postnotes ]
980   {
981     \__postnotes_biblatex_endrefcontext_local:
982     \__postnotes_biblatex_citereset_local:
983   }

```

Restore biblatex variables for each note.

```

984   \AddToHook { postnotes/print/note/begin } [ postnotes ]
985   {
986     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
987     { biblatex@refsection } \l__postnotes_restore_tmp_tl
988     \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
989     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
990     { biblatex@refsegment } \l__postnotes_restore_tmp_tl
991     \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
992     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
993     { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
994     \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
995     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
996     { biblatex@refcontext } \l__postnotes_restore_tmp_tl
997     \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
998   }

```

Auxiliary functions.

__postnotes_biblatex_endrefcontext_local: Replicate the job of \endrefcontext, but with local effects, restrained to the group of \printpostnotes.

```

999   \cs_new_protected:Npn \__postnotes_biblatex_endrefcontext_local:
1000   {
1001     \togglefalse { blx@refcontext }
1002     \tl_clear:N \blx@refcontext@labelprefix
1003     \tl_clear:N \blx@refcontext@labelprefix@real
1004     \tl_set:Nx \blx@refcontext@sortingtemplatename { \blx@sorting }
1005     \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
1006     \tl_set:Nn \blx@refcontext@uniquenametemplatename { global }
1007     \tl_set:Nn \blx@refcontext@labelalphanametemplatename { global }
1008     \blx@edef@refcontext
1009     {
1010       \blx@refcontext@sortingtemplatename /
1011       \blx@refcontext@sortingnamekeytemplatename /
1012       /
1013       \blx@refcontext@uniquenametemplatename /
1014       \blx@refcontext@labelalphanametemplatename
1015     }
1016   }

```

(End of definition for __postnotes_biblatex_endrefcontext_local:.)

`__postnotes_biblatex_citereset_local:` Replicate the job of `\citereset`, but with local effects, restrained to the group of `\printpostnotes`.

```

1017     \cs_new_protected:Npn \__postnotes_biblatex_citereset_local:
1018         {
\global\cslet{blx@bsee@the\c@refsection}\@empty
\global\cslet{blx@fsee@the\c@refsection}\@empty
1019         \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1020         \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
\blx@ibidreset@force
1021         \undef \blx@lastkey@text
1022         \undef \blx@lastkey@foot
\blx@idemreset@force
1023         \undef \blx@lasthash@text
1024         \undef \blx@lasthash@foot
\blx@opcitreset@force
1025         \clist_map_inline:Nn \blx@trackhash@text
1026             { \csundef { blx@lastkey@text@ ##1 } }
1027         \tl_clear:N \blx@trackhash@text
1028         \clist_map_inline:Nn \blx@trackhash@foot
1029             { \csundef { blx@lastkey@foot@ ##1 } }
1030         \tl_clear:N \blx@trackhash@foot
\blx@loccitreset@force
1031         \clist_map_inline:Nn \blx@trackkeys@text
1032             { \csundef { blx@lastnote@text@ ##1 } }
1033         \tl_clear:N \blx@trackkeys@text
1034         \clist_map_inline:Nn \blx@trackkeys@foot
1035             { \csundef { blx@lastnote@foot@ ##1 } }
1036         \tl_clear:N \blx@trackkeys@foot
and all of them do:
1037         \cs_set_eq:NN \blx@lastmpfn \z@
1038     }
(End of definition for \__postnotes_biblatex_citereset_local:.)
1039 }

```

`biblatex`'s `refsections`, contrary to `refsegments` and `refcontexts` which are handled in the \LaTeX side of things (as far as I can tell), need to go through `biber`, and must have correct corresponding citation data written to the `.bcf` file. And the way `\refsection` is implemented presumes each section is only ever begun once (fair...), thus making it difficult to “reopen” it, or append new citations to it later on, when the notes are printed. The start of a `refsection` must be registered on the `.bcf` file, and this is done by `\refsection` (and its auxiliary functions). However, a number of its characteristics make things particularly difficult for the purpose at hand: i) it unconditionally sets a label for the section which, of course, cannot be done twice; and, critically, ii) the optional argument of the environment (which receives the $\langle resources \rangle$) is used to set a local assignment to `\blx@bibfiles`, based on which the relevant information is written to the `.bcf` file, and when the group closes the information is gone. My best attempt is below but it is not good. It feels a wrong approach to “go around”

the intended use of `\refsection` so much, and it can't handle at all its optional argument, for the reasons above. It's also incomplete, since it does not handle restoring `\l__postnotes_biblatex_orig_refsection_tl`.

```

1040 \AddToHook { package/biblatex/after }
1041 {
1042   \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1043   \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1044   \AddToHook { postnotes/print/begin } [ postnotes ]
1045   {
1046     \tl_set:Nx \l__postnotes_biblatex_orig_refsection_tl
1047     { \int_use:N \c@refsection }
1048     \tl_gset:Nx \g__postnotes_biblatex_prev_refsection_tl
1049     \l__postnotes_biblatex_orig_refsection_tl
1050   }
1051   \AddToHook { postnotes/print/note/begin } [ postnotes ]
1052   {
1053     \__postnotes_prop_get:mnN { \l__postnotes_print_note_id_tl }
1054     { biblatex@refsection } \l__postnotes_restore_tmp_tl
1055     \tl_if_eq:NMF
1056     \l__postnotes_restore_tmp_tl
1057     \g__postnotes_biblatex_prev_refsection_tl
1058     {
1059       \int_set:Nn \c@blx@maxsection
1060       { \l__postnotes_restore_tmp_tl - 1 }
1061       \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1062       \l__postnotes_restore_tmp_tl
1063       \group_begin:
1064       \cs_set_eq:NN \label \use_none:n
1065       \cs_set_eq:NN \blx@info \use_none:n
1066       \blx@endrefsection
1067       \refsection
1068       \group_end:
1069     }
1070   }
1071 }
1072 </gobble>

```

zref-user

`\l__postnotes_note_zlabel_tl` Even though the `zlabel` option is provided only when `zref-user` is loaded, `\l__postnotes_note_zlabel_tl` must be unconditionally defined, since it is presumed to exist by `__postnotes_set_user_labels:`.

```

1073 \tl_new:N \l__postnotes_note_zlabel_tl

```

(End of definition for `\l__postnotes_note_zlabel_tl`.)

```

1074 \AddToHook { package/zref-user/after }
1075 {

```

Provide `zlabel` option.

```

1076   \keys_define:mn { postnotes/note }
1077   {
1078     zlabel .tl_set:N = \l__postnotes_note_zlabel_tl ,

```

```

1079         zlabel .value_required:n = true ,
1080     }

```

`\postnotezref` Provide `\postnotezref`.

```

\postnotezref(*){<label>}
1081 \NewDocumentCommand \postnotezref { s m }
1082 { \__postnotes_note_zref:nn {#1} {#2} }

```

(End of definition for \postnotezref.)

`__postnotes_note_zref:nn` The internal version of `\postnotezref`.

```

\__postnotes_note_zref:nn {<star bool>} {<label>}
1083 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1084 {
1085     \group_begin:
1086     \__postnotes_typeset_mark_wrapper:n
1087     {
1088         \bool_lazy_all:nTF
1089         {
1090             { ! #1 }
1091             { \l__postnotes_hyperlink_bool }
1092             { \l__postnotes_zrefhyperref_bool }
1093         }
1094         {
1095             \hyperlink
1096             { \zref@extractdefault {#2} { anchor } { } }
1097             { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1098         }
1099         { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1100     }
1101     \group_end:
1102 }

```

(End of definition for __postnotes_note_zref:nn.)

```

1103 }
1104 \bool_new:N \l__postnotes_zrefhyperref_bool
1105 \AddToHook { package/zref-hyperref/after }
1106 { \bool_set_true:N \l__postnotes_zrefhyperref_bool }

```

zref-clever

```

1107 \AddToHook { package/zref-clever/after }
1108 {
1109     \zcsetup
1110     {
1111         countertype = { postnote = endnote } ,
1112         countertype = { postnotetext = endnote } ,
1113     }
1114 \AddToHook { postnotes/print/begin } [ postnotes ]

```

```

1115     { \zcsetup { counterresetby = { postnotetext = postnotesection } } }
1116   }

```

zref-check

```

1117 \AddToHook { package/zref-check/after }
1118   {
1119     \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1120     {
1121       \AddToHook { postnotes/note/store } [ postnotes ]
1122       {
1123         \prop_gput:cnx { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1124         { zref-check@abschap } { \int_use:N \c@zc@abschap }
1125         \prop_gput:cnx { \__postnotes_data_name:e { \l_postnotes_note_id_tl } }
1126         { zref-check@abssec } { \int_use:N \c@zc@abssec }
1127       }
1128       \AddToHook { postnotes/print/note/begin } [ postnotes ]
1129       {
1130         \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1131         { zref-check@abschap } \l__postnotes_restore_tmp_tl
1132         \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1133         \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
1134         { zref-check@abssec } \l__postnotes_restore_tmp_tl
1135         \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1136       }
1137     }
1138   { }
1139 }

```

amsmath

```

1140 \AddToHook { package/amsmath/after }
1141   {

```

Testing for `\ifmeasuring@` is sufficient to get things right for the measuring passes in math environments.

```

1142     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1143     {
1144       \legacy_if:nT { measuring@ }
1145       {
1146         \bool_set_true:N \l__postnotes_inhibit_note_bool
1147         \bool_set_true:N \l__postnotes_print_plain_mark_bool
1148         \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1149       }
1150     }

```

However, the `\text` macro, defined by `amstext` (required by `amsmath`), poses problems if its own. Despite my best efforts, I could not salvage things from the use of `\mathchoice` and the redefinitions of `\setcounter` and `\addtocounter` performed by `amstext`. Setting `\l__postnotes_maybe_multi_bool` when `firstchoice@` is false grants us a working situation for display style. But the use of `\postnote` inside `\text` (and, if `amsmath` is loaded, `\textnormal`, `\textup`, etc.) in inline math environments is not supported. If a note really needs to be there, one can use the `nomark` option and `\postnoteref`. Things should work in text mode and in display style. For some related discussion with regard to footnotes, see <https://tex.stackexchange.com/a/82820> and, in particular, Barbara Beeton’s comment: “This is certainly bravura code. I do hope it doesn’t result

in a request to add `\footnote` capabilities to `amsmath`'s multi-line display facilities. (The answer will almost certainly be no. We agree with Kopka & Daly.)”

```

1151 \AddToHook { postnotes/note/begin } [ postnotes ]
1152 {
1153   \legacy_if:nF { firstchoice@ }
1154   { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1155 }
1156 }

```

csquotes

```

1157 \AddToHook { package/csquotes/after }
1158 {
1159   \bool_new:N \l__postnotes_csquotes_measuring_bool
1160   \BlockquoteDisable
1161   { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1162   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1163   {
1164     \bool_if:NT \l__postnotes_csquotes_measuring_bool
1165     {
1166       \bool_set_true:N \l__postnotes_inhibit_note_bool
1167       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1168       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1169     }
1170   }
1171 }

```

tabularx

For the identification of the trial passes in `tabularx`, see <https://tex.stackexchange.com/a/640035> (including discussion in the comments, thanks David Carlisle), and also <https://tex.stackexchange.com/a/227155> and <https://tex.stackexchange.com/a/352134>.

```

1172 \AddToHook { package/tabularx/after }
1173 {
1174   \bool_new:N \l__postnotes_tabularx_inside_env_bool
1175   \AddToHook { env/tabularx/begin } [ postnotes ]
1176   {
1177     \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1178     \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1179   }
1180   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1181   {
1182     \bool_lazy_and:nnT
1183     { \l__postnotes_tabularx_inside_env_bool }
1184     { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1185     {
1186       \bool_set_true:N \l__postnotes_inhibit_note_bool
1187       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1188       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1189     }
1190   }
1191 }

```

tabularray

```
1192 \AddToHook { package/tabularray/after }
1193 {
```

Since version 2023A, from 2023-03-01, tabularray offers the `\lTblrMeasuringBool` which is true when measuring and false otherwise. See <https://tex.stackexchange.com/q/675818> and <https://github.com/lvjr/tabularray/issues/179> (thanks Ulrike Fischer).

```
1194   \bool_if_exist:NTF \lTblrMeasuringBool
1195   {
```

I'd be inclined to restrict the inhibition effect to known tabularray environments to “keep things under control”. However this is a dedicated and public boolean, and users can create arbitrary new tabularray environments with `\NewTblrEnviron`, which we either wouldn't catch or have to provide an user interface for. So, for the time being, let's trust this boolean won't be misused by third-parties or users. Note that setting `\l__postnotes_print_plain_mark_stepcounter_bool` to true presumes tabularray's counter module is enabled. But, since this is the only way to get the measuring right in this context if there is more than one `\postnote` inside a given table, `pkgpostnotes` expects and requires the counter module.

```
1196     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1197     {
1198       \bool_if:NT \lTblrMeasuringBool
1199       {
1200         \bool_set_true:N \l__postnotes_inhibit_note_bool
1201         \bool_set_true:N \l__postnotes_print_plain_mark_bool
1202         \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1203       }
1204     }
1205   }
1206 {
```

If the new boolean is not yet available, we use `__postnotes_verify_multipass:N` to distinguish a trial/measure pass from the final one.

```
1207   \clist_map_inline:nn
1208   {
1209     tblr , longtblr , talltblr , booktabs ,
1210     longtabs , talltabs , +array
1211   }
1212   {
1213     \AddToHook { env/#1/begin } [ postnotes ]
1214     { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1215   }
1216 }
1217 }
```

10 Languages

```
\pntitle Set of language specific user variables. They are used in the default value of the heading
\pnhdnotes option and in \pnheaderdefault which, ultimately, is also used in the same place.
\pnhdtopage
\pnhdtopages
```

```
1218 \tl_new:N \pntitle
1219 \tl_new:N \pnhdnotes
1220 \tl_new:N \pnhdtopage
1221 \tl_new:N \pnhdtopages
```

```

1222 \tl_set:Nn \pntitle { Notes }
1223 \tl_set:Nn \pnhdnotes { Notes }
1224 \tl_set:Nn \pnhdtopage { to~page }
1225 \tl_set:Nn \pnhdtopages { to~pages }

```

(End of definition for `\pntitle` and others.)

`__postnotes_define_language:nn` Defines language specific values for *(postnote language)* by storing a set of assignments for the language specific variables in *(setup)*. *(postnote language)* is an internal name, typically the “main” name of the language, based on which we can set specific `babel` or `polyglossia` languages or variants.

```

\__postnotes_define_language:nn {(postnote language)} {(setup)}
1226 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
1227 {
1228   \tl_new:c { g__postnotes_language_ #1 _t1 }
1229   \tl_gset:cn { g__postnotes_language_ #1 _t1 } {#2}
1230 }

```

(End of definition for `__postnotes_define_language:nn`.)

For `babel` we use the new hook system, it’s clean, and avoids the `\addto` pitfalls. The appropriate hook to use is `babel/<language>/beforeextras` so that users can override it with a traditional `\addto\extras<language>`.

Note that, for `babel`, the captions are currently handled in two different ways – the “old way” and the “new way” – and which of them is used depends on the language. Most still use the “old way”, but the problem is that it is not universal. And the “new way” uses a different naming scheme – `\<language><caption>`, which is meant to be set with `\setlocalecaption`, and not suitable for our needs. The `\extras<language>` macros are meant for “arbitrary” code to be run when the language is selected, which is what we want. The captions used to work in the same way, but no longer for languages which use the “new way”.

Note also that there seems to exist some qualms about `babel`’s `\addto`. A number of packages define their own versions of it. Do so at least `varioref` (probably the original), `backref`, and `cleveref`. The latter comments that `\addto` is “flawed”. `babel` itself comments the definition recognizing that there is an “inconsistency”: depending on the case, the operation will be either local or global. This is documented in the manual, which explains this inconsistent behavior is preserved for backward compatibility, and recommends `etoolbox`’s facilities if available. `polyglossia` also recommends `etoolbox`’s `\gappto`. All in all, if there’s need to use the traditional way instead of the new hooks, just rely on `expl3` and use `\tl_gput_right:Nn`.

`__postnotes_set_babel_language:nn` Sets *(babel language)* to execute the setup defined by `__postnotes_define_language:nn` for *(postnote language)* at the `babel/<language>/beforeextras` hook.

```

\__postnotes_set_babel_language:nn {(babel language)} {(postnote language)}
1231 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
1232 {
1233   \ActivateGenericHook { babel/#1/beforeextras }
1234   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
1235   { g__postnotes_language_ #2 _t1 }
1236 }

```

(End of definition for `_postnotes_set_babel_language:nn`.)

`polyglossia` uses a similar set of macros for setting up languages as `babel` does. However, the `\blockextras@⟨language⟩` macros are unfortunately internal (despite what the manual says, that’s what the code does), thus requiring `\makeatletter/\makeatother` for user configuration, which would be an inconvenience. On the other hand, `polyglossia`’s `\captions⟨language⟩` works as in `babel`’s “old way”, meaning it is just a “hook” to which we can append some code. So we use `\captions⟨language⟩` for `polyglossia`. Things may complicate here if there’s need to set up different values for different language variants, since the hooks available are all necessarily internal, but I doubt we’ll ever need variants for these simple strings.

`_postnotes_set_polyglossia_language:mn` Sets `⟨polyglossia language⟩` to execute the setup defined by `_postnotes_define_language:nn` for `⟨postnote language⟩` at the `polyglossia \captions⟨language⟩` hook.

```
\_postnotes_set_polyglossia_language:nn {⟨polyglossia language⟩}
  {⟨postnote language⟩}

1237 \cs_new_protected:Npn \_postnotes_set_polyglossia_language:mn #1#2
1238   {
1239     \AddToHook { package/polyglossia/after }
1240     {
1241       \exp_args:Nnv \csgappto { captions #1 }
1242       { g\_postnotes_language_ #2 _tl }
1243     }
1244   }
```

(End of definition for `_postnotes_set_polyglossia_language:nn`.)

English

```
1245 \_postnotes_define_language:nn { english }
1246   {
1247     \tl_set:Nn \pntitle      { Notes }
1248     \tl_set:Nn \pnhdnotes   { Notes }
1249     \tl_set:Nn \pnhdtopage  { to~page }
1250     \tl_set:Nn \pnhdtopages { to~pages }
1251   }
1252 \_postnotes_set_babel_language:nn { english } { english }
1253 \_postnotes_set_babel_language:nn { british } { english }
1254 \_postnotes_set_babel_language:nn { american } { english }
1255 \_postnotes_set_babel_language:nn { canadian } { english }
1256 \_postnotes_set_babel_language:nn { australian } { english }
1257 \_postnotes_set_babel_language:nn { newzealand } { english }
1258 \_postnotes_set_babel_language:nn { UKenglish } { english }
1259 \_postnotes_set_babel_language:nn { USenglish } { english }
1260 \_postnotes_set_polyglossia_language:nn { english } { english }
```

Portuguese

```
1261 \_postnotes_define_language:nn { portuguese }
1262   {
1263     \tl_set:Nn \pntitle      { Notas }
1264     \tl_set:Nn \pnhdnotes   { Notas }
1265     \tl_set:Nn \pnhdtopage  { da~página }
```



```

1266     \tl_set:Nn \pnhdtopages { das-páginas }
1267   }
1268   \__postnotes_set_babel_language:nn { portuguese } { portuguese }
1269   \__postnotes_set_babel_language:nn { brazilian } { portuguese }
1270   \__postnotes_set_babel_language:nn { portuges } { portuguese }
1271   \__postnotes_set_babel_language:nn { brazil } { portuguese }
1272   \__postnotes_set_polyglossia_language:nn { portuguese } { portuguese }

```

French

French localization validated by ‘Pika78’ at issue [#1](#).

`babel-french` also has `.ldfs` for `francais`, `frenchb`, and `canadien`, but they are deprecated as options and, if used, they fall back to either `french` or `acadian`.

```

1273   \__postnotes_define_language:nn { french }
1274   {
1275     \tl_set:Nn \pntitle { Notes }
1276     \tl_set:Nn \pnhdnotes { Notes }
1277     \tl_set:Nn \pnhdtopage { de-la-page }
1278     \tl_set:Nn \pnhdtopages { des-pages }
1279   }
1280   \__postnotes_set_babel_language:nn { french } { french }
1281   \__postnotes_set_babel_language:nn { acadian } { french }
1282   \__postnotes_set_polyglossia_language:nn { french } { french }

```

German

German localization provided by Herbert Voß at issue [#2](#).

`babel-german` also has `.ldfs` for `germanb` and `ngermanb`, but they are deprecated as options and, if used, they fall back respectively to `german` and `ngerman`.

```

1283   \__postnotes_define_language:nn { german }
1284   {
1285     \tl_set:Nn \pntitle { Endnoten }
1286     \tl_set:Nn \pnhdnotes { Endnoten }
1287     \tl_set:Nn \pnhdtopage { zu-Seite }
1288     \tl_set:Nn \pnhdtopages { zu-Seiten }
1289   }
1290   \__postnotes_set_babel_language:nn { german } { german }
1291   \__postnotes_set_babel_language:nn { ngerman } { german }
1292   \__postnotes_set_babel_language:nn { austrian } { german }
1293   \__postnotes_set_babel_language:nn { naustrian } { german }
1294   \__postnotes_set_babel_language:nn { swissgerman } { german }
1295   \__postnotes_set_babel_language:nn { nswissgerman } { german }
1296   \__postnotes_set_polyglossia_language:nn { german } { german }
1297   </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.

A	<code>\addto</code> 39
<code>\ActivateGenericHook</code>	1233

<code>\addtocounter</code>	36	<code>\cs_new_protected:Npn</code> ..	20, 50, 64, 71, 92, 100, 103, 106, 109, 130, 137, 329, 419, 434, 447, 456, 486, 531, 669, 682, 691, 709, 762, 873, 926, 931, 999, 1017, 1083, 1226, 1231, 1237
<code>\AddToHook</code>	76, 84, 255, 923, 948, 953, 955, 957, 965, 967, 979, 984, 1040, 1044, 1051, 1074, 1105, 1107, 1114, 1117, 1121, 1128, 1140, 1142, 1151, 1157, 1162, 1172, 1175, 1180, 1192, 1196, 1213, 1234, 1239	<code>\cs_new_protected:Npx</code>	74
<code>\AddToHookNext</code>	660	<code>\cs_set:Npn</code>	344, 619, 959, 960
B		<code>\cs_set:Npx</code>	345, 622
<code>\begin</code>	602	<code>\cs_set_eq:NN</code>	177, 194, 550, 1037, 1064, 1065, 1178
<code>\BlockquoteDisable</code>	1160	<code>\csgappto</code>	1241
bool commands:		<code>\csundef</code>	1026, 1029, 1032, 1035
<code>\bool_gset_false:N</code>	661	D	
<code>\bool_gset_true:N</code>	545	<code>\def</code>	3
<code>\bool_if:NTF</code>	30, 260, 349, 395, 399, 415, 423, 494, 543, 548, 601, 651, 684, 928, 1164, 1198	E	
<code>\bool_if_exist:NTF</code>	1194	<code>\end</code>	652
<code>\bool_lazy_all:nTF</code>	1088	<code>\endgroup</code>	21
<code>\bool_lazy_and:nnTF</code>	461, 671, 720, 1182	<code>\endlist</code>	186, 203
<code>\bool_new:N</code>	151, 228, 229, 230, 282, 359, 362, 363, 385, 386, 387, 500, 925, 1104, 1159, 1174	<code>\endnotemark</code>	11
<code>\bool_set_false:N</code>	158, 237, 246, 247, 262, 391, 392, 393	<code>\endnotetext</code>	11
<code>\bool_set_true:N</code>	163, 236, 241, 242, 371, 951, 1106, 1146, 1147, 1148, 1154, 1161, 1166, 1167, 1168, 1177, 1186, 1187, 1188, 1200, 1201, 1202, 1214	<code>\endrefcontext</code>	32
<code>\bool_to_str:N</code>	46	<code>\enoteheading</code>	19
<code>\bool_until_do:nn</code>	553	exp commands:	
box commands:		<code>\exp_args:NNNx</code>	407
<code>\box_use:N</code>	307	<code>\exp_args:Nnv</code>	1234, 1241
<code>\box_wd:N</code>	306	<code>\exp_args:NV</code>	450, 452
<code>\l_tmpa_box</code>	305, 306, 307	<code>\exp_args:Nx</code>	602, 652, 793
C		<code>\exp_not:N</code>	75
<code>\caption</code>	10–12, 30	<code>\exp_not:n</code>	119
<code>\chapter</code>	132	F	
<code>\citereset</code>	33	<code>\fmtversion</code>	5
clist commands:		<code>\footnote</code>	37
<code>\clist_map_inline:Nn</code>	1025, 1028, 1031, 1034	<code>\footnotemark</code>	10–12
<code>\clist_map_inline:nn</code>	1207	<code>\footnotesize</code>	297
<code>\counterwithin</code>	12	<code>\footnotetext</code>	10, 11, 961
cs commands:		fp commands:	
<code>\cs_generate_variant:Nn</code> ..	18, 63, 102, 105, 108, 115, 122, 432, 681, 922	<code>\fp_compare:nNnTF</code>	726
<code>\cs_if_eq_p:NN</code>	1184	<code>\fp_new:N</code>	360
<code>\cs_if_exist:NTF</code>	34, 54, 111, 118, 128, 700, 950	<code>\fp_set:Nn</code>	370
<code>\cs_new:Npn</code>	16, 69, 116, 204	<code>\fp_use:N</code>	31
		G	
		<code>\gappto</code>	39
		group commands:	
		<code>\group_begin:</code>	331, 405, 458, 488, 533, 576, 603, 606, 693, 711, 764, 875, 1063, 1085
		<code>\group_end:</code>	356, 408, 470, 497, 593, 636, 653, 665, 707, 734, 871, 920, 1068, 1101

H	
<code>\hbox</code>	211
hbox commands:	
<code>\hbox_set:Nn</code>	305
<code>\hspace</code>	205
<code>\hyperlink</code>	1095
<code>\hyperref</code>	465
I	
<code>\IfFormatAtLeastTF</code>	5, 6
<code>\IfPackageAtLeastTF</code>	1119
<code>\IfPackageLoadedTF</code>	257
<code>\iftoggle</code>	975
int commands:	
<code>\int_eval:n</code>	929
<code>\int_gincr:N</code>	335, 489, 490, 534
<code>\int_incr:N</code>	406
<code>\int_new:N</code>	322, 485, 516
<code>\int_set:Nn</code>	620, 988, 991, 1059, 1132, 1135
<code>\int_use:N</code>	27, 32, 44, 104, 107, 324, 934, 938, 970, 972, 1019, 1020, 1047, 1124, 1126
iow commands:	
<code>\iow_char:N</code>	482, 483, 668
<code>\iow_now:Nn</code>	80, 88
<code>\iow_shipout_x:Nn</code>	4, 96
<code>\item</code>	21, 23, 686
<code>\itemindent</code>	176, 193
<code>\itemsep</code>	181, 198
K	
keys commands:	
<code>\keys_define:nn</code>	123, 144, 152, 206, 219, 231, 264, 283, 289, 364, 501, 1076
<code>\keys_set:nn</code>	320, 332, 493
L	
<code>\label</code>	12, 450, 1064
<code>\labelsep</code>	205
<code>\labelwidth</code>	175, 192
<code>\leftmargin</code>	174, 191, 192, 193
<code>\leftskip</code>	299
legacy commands:	
<code>\legacy_if:nTF</code>	78, 86, 94, 1144, 1153
<code>\list</code>	172, 189
<code>\listparindent</code>	179, 196
<code>\lTblrMeasuringBool</code>	38, 1194, 1198
M	
<code>\makeatletter</code>	40
<code>\makeatother</code>	40
<code>\makelabel</code>	177, 194
<code>\MakeLinkTarget</code>	2, 346, 626
<code>\mathchoice</code>	10, 36
<code>\mbox</code>	10
<code>\mkbibendnote</code>	31
mode commands:	
<code>\mode_if_horizontal:TF</code>	437, 443
<code>\mode_leave_vertical:</code>	436, 687
msg commands:	
<code>\msg_line_context:</code>	279, 482, 668
<code>\msg_new:nnn</code>	278, 280, 480, 667
<code>\msg_warning:nn</code>	261, 477, 536
<code>\msg_warning:nnn</code>	268, 273
N	
<code>\NeedsTeXFormat</code>	4
<code>\newcounter</code>	321, 528, 529
<code>\NewDocumentCommand</code>	319, 326, 454, 472, 474, 509, 941, 1081
<code>\NewDocumentEnvironment</code>	170, 187
<code>\NewHook</code>	2, 19, 328, 388, 526, 527
<code>\newlabel</code>	4
<code>\NewTblrEnviron</code>	38
<code>\nobreak</code>	440
<code>\noindent</code>	314
<code>\normalfont</code>	205, 211, 305, 315
P	
<code>\PackageError</code>	9
<code>\par</code>	22, 159, 302, 314, 654
<code>\parindent</code>	176, 179, 196, 300
<code>\parsep</code>	180, 197
<code>\parskip</code>	180, 197
<code>\partopsep</code>	183, 200
<code>\pnhdchapfirst</code>	736, 767, 889, 890
<code>\pnhdchaplast</code>	736, 768, 894, 897
<code>\pnhdnamefirst</code>	736, 771, 911, 912
<code>\pnhdnamelast</code>	736, 772, 916, 919
<code>\pnhdnotes</code>	944, 945, 1218, 1248, 1264, 1276, 1286
<code>\pnhdpagefirst</code>	736, 765, 878, 879, 943, 944, 945
<code>\pnhdpagelast</code>	736, 766, 883, 886, 943, 945
<code>\pnhdsectfirst</code>	736, 769, 900, 901
<code>\pnhdsectlast</code>	736, 770, 905, 908
<code>\pnhdtopage</code>	944, 1218, 1249, 1265, 1277, 1287
<code>\pnhdtopages</code>	945, 1218, 1250, 1266, 1278, 1288
<code>\pnheaderdefault</code>	30, 38, 133, 140, 941
<code>\pnheading</code>	6, 125, 128, 538
<code>\pnthechapter</code>	511, 579
<code>\pnthechapternextnote</code>	511, 585
<code>\pnthepage</code>	511, 608
<code>\pnthesection</code>	511, 582
<code>\pnthesectionnextnote</code>	511, 588
<code>\pntitle</code>	132, 139, 1218, 1247, 1263, 1275, 1285

`\postnote` [2](#), [10](#), [12](#),
[13](#), [16](#), [23](#), [24](#), [30](#), [31](#), [36](#), [38](#), [326](#), [959](#)
`\postnotemark` [11](#), [12](#)
`\postnoteref` [12](#), [16](#), [36](#), [454](#)
postnotes commands:
 `\l_postnotes_note_id_tl` [12](#),
 [322](#), [342](#), [346](#), [347](#), [352](#), [354](#), [491](#),
 [495](#), [496](#), [969](#), [971](#), [973](#), [976](#), [1123](#), [1125](#)
 `\l_postnotes_print_note_id_tl`
 [516](#), [556](#), [557](#), [578](#), [581](#), [590](#),
 [609](#), [611](#), [614](#), [617](#), [627](#), [629](#), [631](#),
 [986](#), [989](#), [992](#), [995](#), [1053](#), [1130](#), [1133](#)
postnotes internal commands:
 `\l__postnotes_backlink_bool`
 [230](#), [251](#), [673](#)
 `__postnotes_biblatex_citereset_`
 local: [982](#), [1017](#), [1017](#)
 `__postnotes_biblatex_endrefcontext_`
 local: [981](#), [999](#), [999](#)
 `\l__postnotes_biblatex_orig_`
 refsection_tl . [34](#), [1042](#), [1046](#), [1049](#)
 `\g__postnotes_biblatex_prev_`
 refsection_tl [1043](#), [1048](#), [1057](#), [1061](#)
 `\l__postnotes_clear_queue_seq`
 [22](#), [516](#), [541](#), [662](#)
 `\l__postnotes_csquotes_measuring_`
 bool [1159](#), [1161](#), [1164](#)
 `\l__postnotes_curr_text_page_tl`
 [28](#), [744](#), [786](#),
 [798](#), [799](#), [803](#), [831](#), [834](#), [837](#), [840](#), [851](#)
 `__postnotes_data_name:n` [2](#), [12](#),
 [16](#), [16](#), [18](#), [22](#), [23](#), [24](#), [26](#), [28](#), [36](#), [39](#),
 [41](#), [43](#), [45](#), [47](#), [52](#), [53](#), [56](#), [59](#), [61](#), [66](#),
 [70](#), [72](#), [969](#), [971](#), [973](#), [976](#), [1123](#), [1125](#)
 `__postnotes_define_language:nn`
 [39](#),
 [40](#), [1226](#), [1226](#), [1245](#), [1261](#), [1273](#), [1283](#)
 `__postnotes_extract_pageref:n`
 [5](#), [109](#), [116](#), [122](#), [832](#), [937](#)
 `__postnotes_get_headers_data:N`
 [24](#), [26](#), [28](#), [546](#), [762](#), [762](#)
 `__postnotes_get_pageref:Nn`
 [5](#), [109](#), [109](#), [115](#), [608](#), [797](#), [805](#), [842](#)
 `\g__postnotes_header_chap_first_`
 prop [744](#), [775](#), [833](#), [887](#)
 `\g__postnotes_header_chap_last_`
 prop [744](#), [776](#), [820](#), [861](#), [891](#)
 `\g__postnotes_header_name_first_`
 prop [744](#), [779](#), [839](#), [909](#)
 `\g__postnotes_header_name_last_`
 prop [744](#), [780](#), [826](#), [867](#), [913](#)
 `\g__postnotes_header_page_first_`
 prop [744](#), [773](#), [830](#), [876](#)
 `\g__postnotes_header_page_last_`
 prop [744](#), [774](#), [817](#), [858](#), [880](#)
 `\g__postnotes_header_prev_last_`
 chap_tl [744](#), [782](#), [890](#), [895](#), [897](#)
 `\g__postnotes_header_prev_last_`
 name_tl [744](#), [784](#), [912](#), [917](#), [919](#)
 `\g__postnotes_header_prev_last_`
 page_tl [744](#), [781](#), [879](#), [884](#), [886](#)
 `\g__postnotes_header_prev_last_`
 sect_tl [744](#), [783](#), [901](#), [906](#), [908](#)
 `\g__postnotes_header_sect_first_`
 prop [744](#), [777](#), [836](#), [898](#)
 `\g__postnotes_header_sect_last_`
 prop [744](#), [778](#), [823](#), [864](#), [902](#)
 `\g__postnotes_header_vars_next_`
 bool [29](#), [545](#), [661](#), [925](#), [928](#)
 `\l__postnotes_hyperlink_bool` [228](#),
 [236](#), [241](#), [246](#), [262](#), [423](#), [463](#), [672](#), [1091](#)
 `\l__postnotes_hyperref_warn_bool`
 [229](#), [237](#), [242](#), [247](#), [260](#)
 `__postnotes_inhibit_note:` [389](#)
 `__postnotes_inhibit_note:TF`
 [23](#), [333](#), [385](#)
 `\l__postnotes_inhibit_note_bool`
 [14](#),
 [385](#), [391](#), [415](#), [1146](#), [1166](#), [1186](#), [1200](#)
 `__postnotes_list_makelabel:n`
 [177](#), [194](#), [204](#)
 `__postnotes_make_mark:nnn` [208](#),
 [413](#), [425](#), [429](#), [466](#), [468](#), [1097](#), [1099](#)
 `__postnotes_make_text_mark:nnn`
 [212](#), [675](#), [679](#)
 `\l__postnotes_manual_sortnum_`
 bool [30](#), [362](#), [371](#)
 `\l__postnotes_mark_tl` [25](#), [336](#), [339](#),
 [345](#), [352](#), [358](#), [366](#), [397](#), [402](#), [409](#), [413](#)
 `\l__postnotes_maybe_multi_bool`
 [23](#), [36](#), [46](#), [363](#), [951](#), [1154](#), [1214](#)
 `\l__postnotes_nomark_bool`
 [349](#), [359](#), [380](#)
 `__postnotes_note:nn`
 [13](#), [15](#), [16](#), [327](#), [328](#), [329](#)
 `\g__postnotes_note_id_int`
 [12](#), [322](#), [335](#), [490](#)
 `\l__postnotes_note_label_tl`
 [361](#), [382](#), [449](#), [450](#)
 `__postnotes_note_ref:nn`
 [16](#), [455](#), [456](#), [456](#)
 `\l__postnotes_note_zlabel_tl`
 [34](#), [451](#), [452](#), [1073](#), [1078](#)
 `__postnotes_note_zref:nn`
 [35](#), [1082](#), [1083](#), [1083](#)
 `\l__postnotes_post_printnote_tl`
 [159](#), [218](#), [225](#), [635](#)

<code>\l__postnotes_post_textmark_tl . . .</code>	<code>__postnotes_prop_item:nn</code>
. 217, 223, 686, 689 4, 64, 69, 794, 835, 838, 841
<code>\l__postnotes_pre_textmark_tl . . .</code>	<code>\g__postnotes_queue_seq . . 12, 23,</code>
. 216, 221, 686, 689	24, 26, 322, 341, 491, 535, 541, 542,
<code>\l__postnotes_prev_mark_chap_tl .</code>	544, 546, 553, 555, 561, 567, 637, 643
. 744, 788, 808, 822, 845, 863	<code>\c__postnotes_ref_prefix_tl</code>
<code>\l__postnotes_prev_mark_name_tl .</code> 4, 73, 75, 111, 112, 118, 119, 701
. 744, 790, 812, 828, 849, 869	<code>\l__postnotes_restore_tmp_tl . . .</code>
<code>\l__postnotes_prev_mark_page_tl .</code> 947, 987, 988, 990,
. 744, 787, 806, 819, 843, 860	991, 993, 994, 996, 997, 1054, 1056,
<code>\l__postnotes_prev_mark_sect_tl .</code>	1060, 1062, 1131, 1132, 1134, 1135
. 744, 789, 810, 825, 847, 866	<code>\l__postnotes_saved_spacefactor_-</code>
<code>\l__postnotes_prev_text_page_tl .</code>	tl 433, 439, 444
27, 28, 744, 785, 802, 815, 818, 821,	<code>\g__postnotes_sectid_int 44, 485, 489</code>
824, 827, 850, 856, 859, 862, 865, 868	<code>__postnotes_section:nn</code>
<code>\l__postnotes_print_as_list_bool</code> 17, 473, 485, 486
. . . . 151, 158, 163, 548, 601, 651, 684	<code>\l__postnotes_section_exp_bool . .</code>
<code>\l__postnotes_print_content_tl . .</code> 494, 500, 505
. 516, 591, 592, 618, 634	<code>\g__postnotes_section_name_tl . . .</code>
<code>\l__postnotes_print_counter_tl . .</code> 42, 492, 499, 503
. 516, 615, 621	<code>__postnotes_set_babel_language:nn</code>
<code>\l__postnotes_print_env_tl</code> 39, 1231, 1231, 1252, 1253,
. 150, 160, 164, 602, 652	1254, 1255, 1256, 1257, 1258, 1259,
<code>\l__postnotes_print_format_tl . . .</code>	1268, 1269, 1270, 1271, 1280, 1281,
. 143, 146, 604	1290, 1291, 1292, 1293, 1294, 1295
<code>\l__postnotes_print_mark_tl</code>	<code>__postnotes_set_headers_vars:n .</code>
. 516, 612, 623, 632	. . . 26, 28, 29, 873, 873, 922, 929, 935
<code>\l__postnotes_print_note_id_-</code>	<code>__postnotes_set_headers_vars_-</code>
next_tl 516,	first: 25, 29, 547, 923, 931
563, 568, 570, 584, 587, 639, 644, 646	<code>__postnotes_set_headers_vars_-</code>
<code>__postnotes_print_notes:</code>	next: 25, 29, 923, 924, 926
. 18, 19, 22-24, 510, 531, 531	<code>__postnotes_set_label:nn</code>
<code>\l__postnotes_print_plain_mark_-</code> 5, 92, 92, 101, 104, 107
bool 14,	<code>__postnotes_set_mark_page_-</code>
386, 392, 395, 1147, 1167, 1187, 1201	label:n 5, 24, 92, 100, 102, 347
<code>\l__postnotes_print_plain_mark_-</code>	<code>__postnotes_set_polyglossia_-</code>
stepcounter_bool 14, 38,	language:nn 40,
387, 393, 399, 1148, 1168, 1188, 1202	1237, 1237, 1260, 1272, 1282, 1296
<code>\g__postnotes_print_postnotes_-</code>	<code>__postnotes_set_print_page_-</code>
int 516, 534, 934, 938	label:n 5, 24, 92, 106, 108, 933
<code>\l__postnotes_print_type_curr_tl</code>	<code>__postnotes_set_text_page_-</code>
. 516, 558, 559, 595, 657	label:n 5, 24, 92, 103, 105, 628
<code>\l__postnotes_print_type_next_tl</code>	<code>__postnotes_set_user_labels: . . .</code>
. . . . 516, 564, 571, 573, 640, 647, 649 34, 348, 447, 447
<code>\l__postnotes_print_type_prev_tl</code>	<code>\l__postnotes_sort_bool 282, 285, 543</code>
. 516, 540, 594, 599, 656	<code>\l__postnotes_sort_num_fp 31, 360, 370</code>
<code>__postnotes_prop_gclear:n</code>	<code>__postnotes_sort_queue:N</code>
. 4, 64, 71, 663 24, 544, 709, 709
<code>__postnotes_prop_get:nnN</code>	<code>__postnotes_store:nn . 2, 19, 20, 354</code>
. 4, 64, 64, 557, 569, 577,	<code>__postnotes_store_section:nn . . .</code>
580, 583, 586, 589, 610, 613, 616, 3, 50, 50, 63, 495, 496
645, 697, 714, 715, 718, 719, 724,	<code>\l__postnotes_tabularx_inside_-</code>
725, 807, 809, 811, 844, 846, 848,	env_bool 1174, 1177, 1183
986, 989, 992, 995, 1053, 1130, 1133	

<code>\blx@lasthash@text</code>	1023	<code>\tl_gclear:N</code> 492, 765, 766, 767, 768, 769, 770, 771, 772, 781, 782, 783, 784
<code>\blx@lastkey@foot</code>	1022	<code>\tl_gput_right:Nn</code>
<code>\blx@lastkey@text</code>	1021	39
<code>\blx@lastmpfn</code>	1037	<code>\tl_gset:Nn</code>
<code>\blx@refcontext@context</code>	977	878, 879, 883, 884, 886, 889, 890, 894, 895, 897, 900, 901, 905, 906, 908, 911, 912, 916, 917, 919, 1048, 1229
<code>\blx@refcontext@labelalphanametemplatenam</code>	1007, 1014	<code>\tl_gset_eq:NN</code>
<code>\blx@refcontext@labelprefix</code> ..	1002	1061
<code>\blx@refcontext@labelprefix@real</code>	1003	<code>\tl_if_empty:NTF</code>
<code>\blx@refcontext@sortingnamekeytemplatenam</code>	1005, 1011 336, 397, 449, 451, 799, 815, 856
<code>\blx@refcontext@sortingtemplatenam</code>	1004, 1010	<code>\tl_if_eq:NNTF</code> ... 716, 801, 943, 1055
<code>\blx@refcontext@uniquenametemplatenam</code>	1006, 1013	<code>\tl_if_eq:NnTF</code> 559, 573, 599, 649, 698
<code>\blx@sorting</code>	1004	<code>\tl_if_eq:nTF</code>
<code>\blx@theendnote</code>	959	156, 793
<code>\blx@theendnotetext</code>	960	<code>\tl_new:N</code>
<code>\blx@trackhash@foot</code>	1028, 1030	216, 217, 218, 323, 358, 361, 433, 499, 511, 512, 513, 514, 515, 517, 518, 519, 520, 521, 522, 523, 524, 736, 737, 738, 739, 740, 741, 742, 743, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 947, 1042, 1043, 1073, 1218, 1219, 1220, 1221, 1228
<code>\blx@trackhash@text</code>	1025, 1027	<code>\tl_set:Nn</code>
<code>\blx@trackkeys@foot</code>	1034, 1036	112, 159, 160, 164, 324, 339, 402, 409, 439, 540, 563, 564, 594, 639, 640, 656, 850, 1004, 1005, 1006, 1007, 1046, 1222, 1223, 1224, 1225, 1247, 1248, 1249, 1250, 1263, 1264, 1265, 1266, 1275, 1276, 1277, 1278, 1285, 1286, 1287, 1288
<code>\blx@trackkeys@text</code>	1031, 1033	<code>\l_tmpa_tl</code>
<code>\c@blx@maxsection</code>	1059 697, 698, 714, 716, 718, 721, 724, 726, 877, 878, 881, 883, 884, 888, 889, 892, 894, 895, 899, 900, 903, 905, 906, 910, 911, 914, 916, 917
<code>\c@page</code>	104, 107, 929	<code>\l_tmpb_tl</code> . 715, 716, 719, 722, 725, 726
<code>\c@postnote</code>	15, 27, 32, 406	<code>\togglefalse</code>
<code>\c@postnotetext</code>	620	1001
<code>\c@refsection</code> 970, 988, 1019, 1020, 1047		<code>\toggletrue</code>
<code>\c@refsegment</code>	972, 991	956
<code>\c@z@abschap</code>	1124, 1132	token commands:
<code>\c@z@abssec</code>	1126, 1135	<code>\token_to_str:N</code>
<code>\hyper@linkend</code>	427, 677	81, 89, 97
<code>\hyper@linkstart</code>	426, 676	<code>\topsep</code>
<code>\ifmeasuring@</code>	36	182, 199
<code>\p@postnote</code>	345, 623	
<code>\post@note</code>	4, 73, 81, 89, 97	
<code>\postnotes@required@kernel</code> 3, 4, 6, 11		
<code>\z@</code>	1037	
<code>\zref@extractdefault</code>	1096	
<code>\text</code>	10–12, 36	
<code>\textnormal</code>	36	
<code>\textup</code>	36	
<code>\the</code>	439	
<code>\thechapter</code>	25, 37, 57	
<code>\theHpostnote</code>	12	
<code>\thepage</code>	24, 101	
<code>\thepostnote</code>	15, 339, 402, 409	
<code>\thesection</code>	25, 40, 60	
tl commands:		
<code>\c_empty_tl</code>	120	
<code>\tl_clear:N</code>	67, 113, 785, 786, 787, 788, 789, 790, 1002, 1003, 1019, 1020, 1027, 1030, 1033, 1036	
<code>\tl_const:Nn</code>	73	
		U
		<code>\undef</code>
		1021, 1022, 1023, 1024
		use commands:
		<code>\use:N</code>
		994
		<code>\use_none:n</code>
		1064, 1065
		<code>\UseHook</code>
		48, 343, 394, 539, 607
		W
		<code>\write</code>
		1178, 1184
		Z
		<code>\zcsetup</code>
		1109, 1115
		<code>\zlabel</code>
		452
		<code>\zref</code>
		1097, 1099