

The `postnotes` package*

Code documentation

Gustavo Barros[†]

2022-11-13

Contents

1	Initial setup	2
2	Data	2
3	Options	5
4	<code>\postnote</code>	9
5	<code>\postnoteref</code>	15
6	<code>\postnotesection</code>	16
7	<code>\printpostnotes</code>	17
8	Headers	23
9	Compatibility	29
10	Languages	37
	Index	40

*This file describes v0.1.7, released 2022-11-13.

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

1 Initial setup

Start the DocStrip guards.

```
1 <*package>
   Identify the internal prefix (LATEX3 DocStrip convention).
2 <@@=postnotes>
   Require the new syntax for file/package hooks (ltnews34, ltfilehook).
3 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
4 \IfFormatAtLeastTF{2021-11-15}
5   {}
6   {%
7     \PackageError{postnotes}{LaTeX kernel too old}
8     {%
9       'postnotes' requires a LaTeX kernel 2021-11-15 or newer.%
10      \MessageBreak Loading will abort!%
11     }%
12   \endinput
13 }%
14 \ProvidesExplPackage {postnotes} {2022-11-13} {0.1.7}
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 definition for `__postnotes_data_name:n`.)

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

```
   \__postnotes_store:nn {<note id>} {<note content>}
19 \NewHook { postnotes/store/note }
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/store/note }
49 }

```

(End definition for `__postnotes_store:nn`.)

`__postnotes_store_section:mn` Stores the metadata and $\langle note\ content \rangle$ of `\postnotessection` with ID $\langle note\ id \rangle$, from where it is called.

```

    \__postnotes_store_section:mn { $\langle note\ id \rangle$ } { $\langle note\ content \rangle$ }
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   }

```

(End 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 { $\langle note\ id \rangle$ } { $\langle property \rangle$ } { $\langle tl\ var\ to\ set \rangle$ }
    \__postnotes_prop_item:nn { $\langle note\ id \rangle$ } { $\langle property \rangle$ }
    \__postnotes_prop_gclear:n { $\langle note\ id \rangle$ }
63 \cs_new_protected:Npn \__postnotes_prop_get:nnN #1#2#3
64   {
65     \prop_get:cnNF { \__postnotes_data_name:e {#1} } {#2} #3
66     { \tl_clear:N #3 }
67   }

```

```

68 \cs_new:Npn \__postnotes_prop_item:nn #1#2
69   { \prop_item:cn { \__postnotes_data_name:e {#1} } {#2} }
70 \cs_new_protected:Npn \__postnotes_prop_gclear:n #1
71   { \prop_gclear:c { \__postnotes_data_name:e {#1} } }

```

(End 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>}

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

```

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

`__postnotes_set_mark_page_label:n`
`__postnotes_set_text_page_label:n`
`__postnotes_set_print_page_label:n` 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 {<label name>}
\__postnotes_set_text_page_label:n {<label name>}
\__postnotes_set_print_page_label:n {<label name>}

75 \cs_new_protected:Npn \__postnotes_set_mark_page_label:n #1
76   {
77     \iow_shipout_x:Nn \@auxout
78     { \token_to_str:N \post@note { mark@ #1 } { \thepage } }
79   }
80 \cs_generate_variant:Nn \__postnotes_set_mark_page_label:n { x }
81 \cs_new_protected:Npn \__postnotes_set_text_page_label:n #1
82   {
83     \iow_shipout_x:Nn \@auxout
84     { \token_to_str:N \post@note { text@ #1 } { \int_use:N \c@page } }
85   }
86 \cs_generate_variant:Nn \__postnotes_set_text_page_label:n { x }
87 \cs_new_protected:Npn \__postnotes_set_print_page_label:n #1
88   {
89     \iow_shipout_x:Nn \@auxout
90     { \token_to_str:N \post@note { print@ #1 } { \int_use:N \c@page } }
91   }
92 \cs_generate_variant:Nn \__postnotes_set_print_page_label:n { x }

```

(End definition for `__postnotes_set_mark_page_label:n`, `__postnotes_set_text_page_label:n`, and `__postnotes_set_print_page_label:n`.)

`__postnotes_get_pageref:Nn`
`__postnotes_extract_pageref:n` Reference data extraction functions.

```

\__postnotes_get_pageref:Nn {<tl var to set>} {<label name>}
\__postnotes_extract_pageref:n {<label name>}

```

```

93 \cs_new_protected:Npn \__postnotes_get_pageref:Nn #1#2
94 {
95   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #2 }
96     { \tl_set:Nv #1 { \c__postnotes_ref_prefix_tl @ #2 } }
97     { \tl_clear:N #1 }
98 }
99 \cs_generate_variant:Nn \__postnotes_get_pageref:Nn { Nx }
100 \cs_new:Npn \__postnotes_extract_pageref:n #1
101 {
102   \cs_if_exist:cTF { \c__postnotes_ref_prefix_tl @ #1 }
103     { \exp_not:v { \c__postnotes_ref_prefix_tl @ #1 } }
104     { \c_empty_tl }
105 }
106 \cs_generate_variant:Nn \__postnotes_extract_pageref:n { e }

```

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

3 Options

heading option

```

107 \keys_define:nn { postnotes/setup }
108 {
109   heading .cs_set_protected:Np = \pnheading ,
110   heading .value_required:n = true ,
111 }

```

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

```

112 \cs_if_exist:cTF { chapter }
113 {
114   \cs_new_protected:Npn \pnheading
115     {
116       \chapter*{\pntitle}
117       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
118     }
119 }
120 {
121   \cs_new_protected:Npn \pnheading
122     {
123       \section*{\pntitle}
124       \@mkboth{\pnheaderdefault}{\pnheaderdefault}
125     }
126 }

```

(End definition for `\pnheading`.)

format option

```

127 \tl_new:N \l__postnotes_print_format_tl
128 \keys_define:nn { postnotes/setup }
129 {
130   format .tl_set:N = \l__postnotes_print_format_tl ,
131   format .initial:n = { \small } ,

```

```

132     format .value_required:n = true ,
133   }

```

listenv option

```

134 \tl_new:N \l__postnotes_print_env_tl
135 \bool_new:N \l__postnotes_print_as_list_bool
136 \keys_define:nn { postnotes/setup }
137   {
138     listenv .code:n =
139       {
140         \tl_if_eq:nnTF {#1} { none }
141           {
142             \bool_set_false:N \l__postnotes_print_as_list_bool
143             \tl_set:Nn \l__postnotes_post_printnote_tl { \par }

```

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

```

144             \tl_set:Nn \l__postnotes_print_env_tl { itemize }
145           }
146         {
147             \bool_set_true:N \l__postnotes_print_as_list_bool
148             \tl_set:Nn \l__postnotes_print_env_tl {#1}
149           }
150       } ,
151     listenv .initial:n = { postnoteslist } ,
152     listenv .value_required:n = true ,
153   }

```

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*).

```

154 \NewDocumentEnvironment { postnoteslist } { }
155   {
156     \list { }
157     {
158       \setlength { \leftmargin } { 0pt }
159       \setlength { \labelwidth } { 0pt }
160       \setlength { \itemindent } { .5\parindent }
161       \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n
162       \setlength { \rightmargin } { 0pt }
163       \setlength { \listparindent } { \parindent }
164       \setlength { \parsep } { \parskip }
165       \setlength { \itemsep } { 0pt }
166       \setlength { \topsep } { .5\topsep }
167       \setlength { \partopsep } { .5\partopsep }
168     }
169   }
170 \endlist }
171 \NewDocumentEnvironment { postnoteslisthang } { }
172   {
173     \list { }
174     {
175       \setlength { \leftmargin } { 1em }
176       \setlength { \labelwidth } { -\leftmargin }
177       \setlength { \itemindent } { -2\leftmargin }
178       \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n

```

```

179     \setlength { \rightmargin } { Opt }
180     \setlength { \listparindent } { \parindent }
181     \setlength { \parsep } { \parskip }
182     \setlength { \itemsep } { Opt }
183     \setlength { \topsep } { .5\topsep }
184     \setlength { \partopsep } { .5\partopsep }
185   }
186 }
187 { \endlist }
188 \cs_new:Npn \__postnotes_list_makelabel:n #1
189 { \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.

```

190 \keys_define:nn { postnotes/setup }
191 {
192   makemark .cs_set:Np = \__postnotes_make_mark:nnn #1#2#3 ,
193   makemark .value_required:n = true ,

```

From the default kernel definition of \@makefnmark.

```

194   makemark .initial:n =
195     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
196   maketextmark .cs_set:Np = \__postnotes_make_text_mark:nnn #1#2#3 ,
197   maketextmark .value_required:n = true ,
198   maketextmark .initial:n = { #2 #1 . #3 } ,
199 }

```

pretextmark, posttextmark, postprintnote options

```

200 \tl_new:N \l__postnotes_pre_textmark_tl
201 \tl_new:N \l__postnotes_post_textmark_tl
202 \tl_new:N \l__postnotes_post_printnote_tl
203 \keys_define:nn { postnotes/setup }
204 {
205   pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
206   pretextmark .value_required:n = true ,
207   posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
208   posttextmark .value_required:n = true ,
209   postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
210   postprintnote .value_required:n = true ,
211 }

```

hyperref and backlink options

```

212 \bool_new:N \l__postnotes_hyperlink_bool
213 \bool_new:N \l__postnotes_hyperref_warn_bool
214 \bool_new:N \l__postnotes_backlink_bool
215 \keys_define:nn { postnotes/setup }
216 {
217   hyperref .choice: ,
218   hyperref / auto .code:n =
219     {
220       \bool_set_true:N \l__postnotes_hyperlink_bool

```

```

221     \bool_set_false:N \l__postnotes_hyperref_warn_bool
222   } ,
223   hyperref / true .code:n =
224   {
225     \bool_set_true:N \l__postnotes_hyperlink_bool
226     \bool_set_true:N \l__postnotes_hyperref_warn_bool
227   } ,
228   hyperref / false .code:n =
229   {
230     \bool_set_false:N \l__postnotes_hyperlink_bool
231     \bool_set_false:N \l__postnotes_hyperref_warn_bool
232   } ,
233   hyperref .initial:n = auto ,
234   hyperref .default:n = true ,
235   backlink .bool_set:N = \l__postnotes_backlink_bool ,
236   backlink .initial:n = true ,
237   backlink .default:n = true ,
238 }
239 \AddToHook { begindocument }
240 {
241   \IfPackageLoadedTF { hyperref }
242   { }
243   {
244     \bool_if:NT \l__postnotes_hyperref_warn_bool
245     { \msg_warning:nn { postnotes } { missing-hyperref } }
246     \bool_set_false:N \l__postnotes_hyperlink_bool
247   }
248   \keys_define:nn { postnotes/setup }
249   {
250     hyperref .code:n =
251     {
252       \msg_warning:nnn { postnotes }
253       { option-preamble-only } { hyperref }
254     } ,
255     backlink .code:n =
256     {
257       \msg_warning:nnn { postnotes }
258       { option-preamble-only } { backlink }
259     } ,
260   }
261 }
262 \msg_new:nnn { postnotes } { option-preamble-only }
263 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
264 \msg_new:nnn { postnotes } { missing-hyperref }
265 { Missing~'hyperref'~package.~Setting~'hyperref=false'. }

```

sort option

```

266 \bool_new:N \l__postnotes_sort_bool
267 \keys_define:nn { postnotes/setup }
268 {
269   sort .bool_set:N = \l__postnotes_sort_bool ,
270   sort .initial:n = true ,
271   sort .default:n = true ,
272 }

```


style option

```
273 \keys_define:nn { postnotes/setup }
274 {
275   style .choice: ,
276   style / endnotes .meta:n =
277   {
278     listenv = none ,
279     format =
280     {
281       \footnotesize
282       \setlength { \rightskip } { 0pt }
283       \setlength { \leftskip } { 0pt }
284       \setlength { \parindent } { 1.8em }
285     } ,
286     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.

```
287     maketextmark =
288     {
289       \hbox_set:Nn \l_tmpa_box { \@textsuperscript { \normalfont ##1 } }
290       \skip_horizontal:n { - \box_wd:N \l_tmpa_box }
291       ##2 \box_use:N \l_tmpa_box ##3
292     } ,
293   } ,
294   style / pagenote .meta:n =
295   {
296     listenv = none ,
297     format = { } ,
298     pretextmark = { \par\noindent } ,
299     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
300     posttextmark = { ~ } ,
301   } ,
302 }
```

\postnotesetup

\postnotesetup Provide \postnotesetup.

```
\postnotesetup{options}
303 \NewDocumentCommand \postnotesetup { m }
304 { \keys_set:nn { postnotes/setup } {#1} }
```

(End 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 has 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 be 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.

```
305 \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 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 convenience variable storing the counter’s value. `\g__postnotes_queue_seq` stores the sequence of notes’ IDs that to be processed by the next call of `\printpostnotes`.

```
306 \int_new:N \g__postnotes_note_id_int
307 \tl_new:N \l__postnotes_note_id_tl
308 \tl_set:Nn \l__postnotes_note_id_tl { \int_use:N \g__postnotes_note_id_int }
309 \seq_new:N \g__postnotes_queue_seq
```

(End definition for `\g__postnotes_note_id_int`, `\l__postnotes_note_id_tl`, and `\g__postnotes_queue_seq`.)

`\postnote` Provide `\postnote`.

```
\postnote [options] {note text}
```

```
310 \NewDocumentCommand \postnote { 0 { } +m }
311 { \__postnotes_note:nn {#1} {#2} }
```

(End 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}
```

```
312 \NewHook { postnotes/note/begin }
313 \cs_new_protected:Npn \__postnotes_note:nn #1#2
314 {
315   \group_begin:
316   \keys_set:nn { postnotes/note } {#1}
317   \__postnotes_inhibit_note:F
318   {
319     \int_gincr:N \g__postnotes_note_id_int
320     \tl_if_empty:NT \l__postnotes_mark_tl
321     {
322       \stepcounter { postnote }
```

```

323     \tl_set:Nx \l__postnotes_mark_tl { \thepostnote }
324   }
325   \seq_gput_right:Nx \g__postnotes_queue_seq
326     { \l__postnotes_note_id_tl }
327   \UseHook { postnotes/note/begin }
328   \cs_set:Npn \@currentcounter { postnote }
329   \cs_set:Npx \@currentlabel { \p@postnote \l__postnotes_mark_tl }
330   \__postnotes_hyperref_make_currentHref:n
331     { postnote. \l__postnotes_note_id_tl .mark }
332   \__postnotes_set_mark_page_label:x { \l__postnotes_note_id_tl }
333   \__postnotes_set_user_labels:
334   \bool_if:NTF \l__postnotes_nomark_bool
335     {
336     \bool_if:NT \l__postnotes_hyperlink_bool
337       {
338         \__postnotes_hyperref_set_anchor:n
339           { postnote. \l__postnotes_note_id_tl .mark }
340       }
341     }
342     {
343     \__postnotes_typeset_mark:xV
344       { \l__postnotes_note_id_tl } \l__postnotes_mark_tl
345     }
346   \__postnotes_store:nn { \l__postnotes_note_id_tl } {#2}
347   }
348   \group_end:
349 }

```

(End definition for `__postnotes_note:mn`.)

Options for `\postnote`.

```

350 \tl_new:N \l__postnotes_mark_tl
351 \bool_new:N \l__postnotes_nomark_bool
352 \fp_new:N \l__postnotes_sort_num_fp
353 \tl_new:N \l__postnotes_note_label_tl
354 \bool_new:N \l__postnotes_manual_sortnum_bool
355 \bool_new:N \l__postnotes_maybe_multi_bool
356 \keys_define:nn { postnotes/note }
357 {
358   mark .tl_set:N = \l__postnotes_mark_tl ,
359   mark .value_required:n = true ,
360   nomark .bool_set:N = \l__postnotes_nomark_bool ,
361   nomark .default:n = true ,
362   sortnum .code:n =
363     {
364       \fp_set:Nn \l__postnotes_sort_num_fp {#1}
365       \bool_set_true:N \l__postnotes_manual_sortnum_bool
366     } ,
367   sortnum .value_required:n = true ,
368   label .tl_set:N = \l__postnotes_note_label_tl ,
369   label .value_required:n = true ,
370 }

```

`__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` and `\l__postnotes_print_plain_mark_bool` as appropriate to the case.

```

371 \bool_new:N \l__postnotes_inhibit_note_bool
372 \bool_new:N \l__postnotes_print_plain_mark_bool
373 \NewHook { postnotes/note/inhibit }
374 \prg_new_protected_conditional:Npnn \__postnotes_inhibit_note: { F }
375 {
376   \bool_set_false:N \l__postnotes_inhibit_note_bool
377   \bool_set_false:N \l__postnotes_print_plain_mark_bool
378   \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. Since we’d only print this mark in case of inhibition, when we don’t actually step the counter, 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.

```

379   \bool_if:NT \l__postnotes_print_plain_mark_bool
380     {
381       \tl_if_empty:NT \l__postnotes_mark_tl
382         {
383           \group_begin:
384           \int_incr:N \c@postnote
385           \exp_args:NNNx
386           \group_end:
387           \tl_set:Nn \l__postnotes_mark_tl { \thepostnote }
388         }
389       \__postnotes_typeset_mark_wrapper:n
390       { \__postnotes_make_mark:nnn { \l__postnotes_mark_tl } { } { } }
391     }
392   \bool_if:NTF \l__postnotes_inhibit_note_bool
393     { \prg_return_true: }
394     { \prg_return_false: }
395 }

```

(End 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>}

396 \cs_new_protected:Npn \__postnotes_typeset_mark:nn #1#2
397 {
398   \__postnotes_typeset_mark_wrapper:n
399   {
400     \bool_if:NTF \l__postnotes_hyperlink_bool
401       {
402         \__postnotes_hyperref_set_anchor:n { postnote. #1 .mark }
403         \__postnotes_make_mark:nnn {#2}

```

```

404         { \hyper@linkstart { link } { postnote. #1 .text } }
405         { \hyper@linkend }
406     }
407     { \__postnotes_make_mark:nnn {#2} { } { } }
408 }
409 }
410 \cs_generate_variant:Nn \__postnotes_typeset_mark:nn { xV }
411 \tl_new:N \l__postnotes_saved_spacefactor_tl
412 \cs_new_protected:Npn \__postnotes_typeset_mark_wrapper:n #1
413 {
414     \mode_leave_vertical:
415     \mode_if_horizontal:T
416     {
417         \tl_set:Nx \l__postnotes_saved_spacefactor_tl { \the\spacefactor }
418         \nobreak
419     }
420     #1
421     \mode_if_horizontal:T
422     { \spacefactor \l__postnotes_saved_spacefactor_tl }
423     \scan_stop:
424 }

```

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

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

```

425 \cs_new_protected:Npn \__postnotes_set_user_labels:
426 {
427     \tl_if_empty:NF \l__postnotes_note_label_tl
428     { \exp_args:NV \label \l__postnotes_note_label_tl }
429     \tl_if_empty:NF \l__postnotes_note_zlabel_tl
430     { \exp_args:NV \zlabel \l__postnotes_note_zlabel_tl }
431 }

```

(End definition for `__postnotes_set_user_labels:.`)

5 `\postnoteref`

`\postnoteref` Provide `\postnoteref`.

```

\postnoteref(*){\label}

432 \NewDocumentCommand \postnoteref { s m }
433 { \__postnotes_note_ref:nn {#1} {#2} }

```

(End definition for `\postnoteref`.)

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

```

\__postnotes_note_ref:nn {<star bool>} {\label}

```

```

434 \cs_new_protected:Npn \__postnotes_note_ref:nn #1#2
435 {
436   \group_begin:
437   \__postnotes_typeset_mark_wrapper:n
438   {
439     \bool_lazy_and:nnTF
440     { ! #1 }
441     { \l__postnotes_hyperlink_bool }
442     {
443       \hyperref [#2]
444       { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
445     }
446     { \__postnotes_make_mark:nnn { \__postnotes_ref_star:n {#2} } { } { } }
447   }
448   \group_end:
449 }

```

(End definition for __postnotes_note_ref:nn.)

6 \postnotesection

\postnotesection Provide \postnotesection and \postnotesectionx.
\postnotesectionx

```

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

450 \NewDocumentCommand \postnotesection { 0 { } +m }
451 { \__postnotes_section:nn {#1} {#2} }
452 \NewDocumentCommand \postnotesectionx { 0 { } +m }
453 { \__postnotes_section:nx {#1} {#2} }

```

(End definition for \postnotesection and \postnotesectionx.)

__postnotes_section:nn The internal version of \postnotesection.

```

\__postnotes_section:nn {<options>} {<content>}

454 \int_new:N \g__postnotes_sectid_int
455 \cs_new_protected:Npn \__postnotes_section:nn #1#2
456 {
457   \group_begin:
458   \int_gincr:N \g__postnotes_sectid_int
459   \int_gincr:N \g__postnotes_note_id_int
460   \seq_gput_right:Nx \g__postnotes_queue_seq { \l__postnotes_note_id_tl }
461   \tl_gclear:N \g__postnotes_section_name_tl
462   \keys_set:nn { postnotes/section } {#1}
463   \__postnotes_store_section:nn { \l__postnotes_note_id_tl } {#2}
464   \group_end:
465 }
466 \cs_generate_variant:Nn \__postnotes_section:nn { nx }

```


(End definition for `_postnotes_section:nn`.)

Options for `\postnotessection`. 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.

```
467 \tl_new:N \g__postnotes_section_name_tl
468 \keys_define:nn { postnotes/section }
469 {
470   name .tl_gset:N = \g__postnotes_section_name_tl ,
471   name .value_required:n = true ,
472 }
```

7 `\printpostnotes`

`\printpostnotes` Provide `\printpostnotes`.

`\printpostnotes`

```
473 \NewDocumentCommand \printpostnotes { }
474 { \_postnotes_print_notes: }
```

(End definition for `\printpostnotes`.)

`\pnthechapter` User facing variables, aimed at making available some of the notes’ and sections’ metadata for the user at specific contexts.

```
\pnthechapternextnote 475 \tl_new:N \pnthechapter
\pnthesectionnextnote 476 \tl_new:N \pnthesection
\pnthepage              477 \tl_new:N \pnthechapternextnote
                        478 \tl_new:N \pnthesectionnextnote
                        479 \tl_new:N \pnthepage
```

(End definition for `\pnthechapter` and others.)

`\g__postnotes_print_postnotes_int` Auxiliary variables for `_postnotes_print_notes:`.

```
\l__postnotes_print_note_id_tl 480 \int_new:N \g__postnotes_print_postnotes_int
\l__postnotes_print_note_id_next_tl 481 \tl_new:N \l__postnotes_print_note_id_tl
\l__postnotes_print_counter_tl 482 \tl_new:N \l__postnotes_print_note_id_next_tl
\l__postnotes_print_mark_tl 483 \tl_new:N \l__postnotes_print_counter_tl
\l__postnotes_print_type_curr_tl 484 \tl_new:N \l__postnotes_print_mark_tl
\l__postnotes_print_type_next_tl 485 \tl_new:N \l__postnotes_print_type_curr_tl
\l__postnotes_print_type_prev_tl 486 \tl_new:N \l__postnotes_print_type_next_tl
\l__postnotes_print_content_tl 487 \tl_new:N \l__postnotes_print_type_prev_tl
\l__postnotes_clear_queue_seq 488 \tl_new:N \l__postnotes_print_content_tl
489 \seq_new:N \l__postnotes_clear_queue_seq
```

(End definition for `\g__postnotes_print_postnotes_int` and others.)

`_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/eachnote` 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.

```
490 \NewHook { postnotes/print/begin }
491 \NewHook { postnotes/print/eachnote }
```

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 `postnotessection` 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 `postnotessection` 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, 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 `postnotessection`. 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.

```
492 \newcounter { postnotetext }
493 \newcounter { postnotessection }
494 \setcounter { postnotessection } { 10000 }
```

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

```

__postnotes_print_notes:
495 \cs_new_protected:Npn \__postnotes_print_notes:
496 {
497   \group_begin:
498   \int_gincr:N \g__postnotes_print_postnotes_int
499   \seq_if_empty:NTF \g__postnotes_queue_seq
500     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
501   {
502     \pnheading
503     \UseHook { postnotes/print/begin }
504     \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
505     \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_queue_seq
506     \__postnotes_verify_multipass:N \g__postnotes_queue_seq
507     \bool_if:NT \l__postnotes_sort_bool
508       { \__postnotes_sort_queue:N \g__postnotes_queue_seq }
509     \bool_gset_true:N \g__postnotes_header_vars_next_bool
510     \__postnotes_get_headers_data:N \g__postnotes_queue_seq
511     \__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](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).

```

512     \bool_if:NF \l__postnotes_print_as_list_bool
513     {
514         \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
515         \@afterindenttrue
516     }
517 \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_queue_seq }
518 {
519     \seq_gpop_left:NN \g__postnotes_queue_seq
520     \l__postnotes_print_note_id_tl
521     \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
522     { type } \l__postnotes_print_type_curr_tl
523     \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
524     { % type_curr = 'section'
525         \seq_if_empty:NTF \g__postnotes_queue_seq
526         {
527             \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
528             \tl_set:Nn \l__postnotes_print_type_next_tl { close }
529         }
530         {
531             \seq_get_left:NN \g__postnotes_queue_seq
532             \l__postnotes_print_note_id_next_tl
533             \__postnotes_prop_get:nnN
534             { \l__postnotes_print_note_id_next_tl }
535             { type } \l__postnotes_print_type_next_tl
536         }
537     }

```

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

```

537     \tl_if_eq:NnT \l__postnotes_print_type_next_tl { note }
538     {
539         \stepcounter { postnotessection }
540         \group_begin:
541         \__postnotes_prop_get:nnN
542         { \l__postnotes_print_note_id_tl }
543         { thechapter } \pnthechapter
544         \__postnotes_prop_get:nnN
545         { \l__postnotes_print_note_id_tl }
546         { thesection } \pnthesection
547         \__postnotes_prop_get:nnN
548         { \l__postnotes_print_note_id_next_tl }
549         { thechapter } \pnthechapternextnote
550         \__postnotes_prop_get:nnN
551         { \l__postnotes_print_note_id_next_tl }
552         { thesection } \pnthesectionnextnote
553         \__postnotes_prop_get:nnN
554         { \l__postnotes_print_note_id_tl }
555         { content } \l__postnotes_print_content_tl
556         \l__postnotes_print_content_tl
557         \group_end:

```

Set `type_prev` for the next iteration.

```

558         \tl_set:NV \l__postnotes_print_type_prev_tl
559         \l__postnotes_print_type_curr_tl
560     }

```

```

561 }
562 { % type_curr = 'note'
563 \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }
564 {
565     \bool_if:NTF \l__postnotes_print_as_list_bool
566     { \exp_args:Nx \begin { \l__postnotes_print_env_tl } }
567     { \group_begin: }
568     \l__postnotes_print_format_tl
569 }
570 \group_begin:
571 \UseHook { postnotes/print/eachnote }
572 \__postnotes_get_pageref:Nx \pnthepage
573 { mark@ \l__postnotes_print_note_id_tl }
574 \__postnotes_prop_get:nnN
575 { \l__postnotes_print_note_id_tl }
576 { mark } \l__postnotes_print_mark_tl
577 \__postnotes_prop_get:nnN
578 { \l__postnotes_print_note_id_tl }
579 { counter } \l__postnotes_print_counter_tl
580 \__postnotes_prop_get:nnN
581 { \l__postnotes_print_note_id_tl }
582 { content } \l__postnotes_print_content_tl
583 \cs_set:Npn \@currentcounter { postnotetext }
584 \int_set:Nn \c@postnotetext
585 { \l__postnotes_print_counter_tl }
586 \cs_set:Npx \@currentlabel
587 { \p@postnote \l__postnotes_print_mark_tl }
588 \__postnotes_hyperref_make_currentHref:n
589 { postnote. \l__postnotes_print_note_id_tl .text }
590 \__postnotes_text_mark_wrapper:n
591 {
592     \__postnotes_set_text_page_label:x
593     { \l__postnotes_print_note_id_tl }
594     \__postnotes_typeset_text_mark:eV
595     { \l__postnotes_print_note_id_tl }
596     \l__postnotes_print_mark_tl
597 }
598 \l__postnotes_print_content_tl
599 \l__postnotes_post_printnote_tl
600 \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.

```

601 \seq_if_empty:NTF \g__postnotes_queue_seq
602 {
603     \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
604     \tl_set:Nn \l__postnotes_print_type_next_tl { close }
605 }
606 {
607     \seq_get_left:NN \g__postnotes_queue_seq
608     \l__postnotes_print_note_id_next_tl
609     \__postnotes_prop_get:nnN
610     { \l__postnotes_print_note_id_next_tl }

```

```

611         { type } \l__postnotes_print_type_next_tl
612     }
613     \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
614     {
615         \bool_if:NTF \l__postnotes_print_as_list_bool
616         { \exp_args:Nx \end { \l__postnotes_print_env_tl } }
617         { \group_end: }
618     }

```

Set `type_prev` for the next iteration.

```

619         \tl_set:NV \l__postnotes_print_type_prev_tl
620         \l__postnotes_print_type_curr_tl
621     }
622 }
623 \AddToHookNext { shipout/after }
624 { \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.

```

625     \seq_map_inline:Nn \l__postnotes_clear_queue_seq
626     { \__postnotes_prop_gc_clear:n { ##1 } }
627 }
628 \group_end:
629 }

```

(End definition for `__postnotes_print_notes:`)

```

630 \msg_new:nnn { postnotes } { empty-printpostnotes }
631 { 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>}

632 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
633 {
634     \bool_if:NTF \l__postnotes_hyperlink_bool
635     {
636         \__postnotes_hyperref_set_anchor:n { postnote. #1 .text }
637         \bool_if:NTF \l__postnotes_backlink_bool
638         {
639             \__postnotes_make_text_mark:nnn {#2}
640             { \hyper@linkstart { link } { postnote. #1 .mark } }
641             { \hyper@linkend }
642         }
643         { \__postnotes_make_text_mark:nnn {#2} { } { } }
644     }
645     { \__postnotes_make_text_mark:nnn {#2} { } { } }
646 }
647 \cs_generate_variant:Nn \__postnotes_typeset_text_mark:nn { eV }
648 \cs_new_protected:Npn \__postnotes_text_mark_wrapper:n #1
649 {

```

```

650     \bool_if:NTF \l__postnotes_print_as_list_bool
651     {
652         \item
653         [ \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl ]
654     }
655     { \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl }
656 }

```

(End 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)
657 \cs_new_protected:Npn \__postnotes_verify_multipass:N #1
658 {
659     \group_begin:
660     \seq_clear:N \l_tmpa_seq
661     \seq_map_inline:Nn #1
662     {
663         \__postnotes_prop_get:nnN {##1} { multibool } \l_tmpa_tl
664         \tl_if_eq:NnTF \l_tmpa_tl { true }
665         {
666             \cs_if_exist:cT
667             { \c__postnotes_ref_prefix_tl @ mark@ ##1 }
668             { \seq_put_right:Nn \l_tmpa_seq {##1} }
669         }
670         { \seq_put_right:Nn \l_tmpa_seq {##1} }
671     }
672     \seq_gset_eq:NN #1 \l_tmpa_seq
673     \group_end:
674 }

```

(End definition for `__postnotes_verify_multipass:N`.)

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

```

\__postnotes_sort_queue:N (\g__postnotes_queue_seq)
675 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
676 {
677   \group_begin:
678   \seq_gsort:Nn #1
679   {
680     \__postnotes_prop_get:nnN {##1} { pnsectid } \l_tmpa_tl
681     \__postnotes_prop_get:nnN {##2} { pnsectid } \l_tmpb_tl
682     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
683     {
684       \__postnotes_prop_get:nnN {##1} { type } \l_tmpa_tl
685       \__postnotes_prop_get:nnN {##2} { type } \l_tmpb_tl
686       \bool_lazy_and:nnTF
687       { \str_if_eq_p:Vn \l_tmpa_tl { note } }
688       { \str_if_eq_p:Vn \l_tmpb_tl { note } }
689       {
690         \__postnotes_prop_get:nnN {##1} { sortnum } \l_tmpa_tl
691         \__postnotes_prop_get:nnN {##2} { sortnum } \l_tmpb_tl
692         \fp_compare:nNnTF { \l_tmpa_tl } > { \l_tmpb_tl }
693         { \sort_return_swapped: }
694         { \sort_return_same: }
695       }
696       { \sort_return_same: }
697     }
698     { \sort_return_same: }
699   }
700   \group_end:
701 }

```

(End 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>	702 <code>\tl_new:N \pnhdpagefirst</code>
<code>\pnhdsectfirst</code>	703 <code>\tl_new:N \pnhdpagelast</code>
<code>\pnhdsectlast</code>	704 <code>\tl_new:N \pnhdchapfirst</code>
<code>\pnhdnamefirst</code>	705 <code>\tl_new:N \pnhdchaplast</code>
<code>\pnhdnamelast</code>	706 <code>\tl_new:N \pnhdsectfirst</code>
	707 <code>\tl_new:N \pnhdsectlast</code>
	708 <code>\tl_new:N \pnhdnamefirst</code>
	709 <code>\tl_new:N \pnhdnamelast</code>

(End 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>	710 <code>\prop_new:N \g__postnotes_header_page_first_prop</code>
<code>\g__postnotes_header_chap_first_prop</code>	711 <code>\prop_new:N \g__postnotes_header_page_last_prop</code>
<code>\g__postnotes_header_chap_last_prop</code>	712 <code>\prop_new:N \g__postnotes_header_chap_first_prop</code>
<code>\g__postnotes_header_sect_first_prop</code>	713 <code>\prop_new:N \g__postnotes_header_chap_last_prop</code>
<code>\g__postnotes_header_sect_last_prop</code>	714 <code>\prop_new:N \g__postnotes_header_sect_first_prop</code>
<code>\g__postnotes_header_name_first_prop</code>	715 <code>\prop_new:N \g__postnotes_header_sect_last_prop</code>
<code>\g__postnotes_header_name_last_prop</code>	716 <code>\prop_new:N \g__postnotes_header_name_first_prop</code>
<code>\g__postnotes_header_prev_last_page_tl</code>	717 <code>\prop_new:N \g__postnotes_header_name_last_prop</code>
<code>\g__postnotes_header_prev_last_chap_tl</code>	718 <code>\tl_new:N \g__postnotes_header_prev_last_page_tl</code>
<code>\g__postnotes_header_prev_last_sect_tl</code>	719 <code>\tl_new:N \g__postnotes_header_prev_last_chap_tl</code>
<code>\g__postnotes_header_prev_last_name_tl</code>	720 <code>\tl_new:N \g__postnotes_header_prev_last_sect_tl</code>
<code>\l__postnotes_prev_text_page_tl</code>	721 <code>\tl_new:N \g__postnotes_header_prev_last_name_tl</code>
<code>\l__postnotes_curr_text_page_tl</code>	722 <code>\tl_new:N \l__postnotes_prev_text_page_tl</code>
<code>\l__postnotes_prev_mark_page_tl</code>	723 <code>\tl_new:N \l__postnotes_curr_text_page_tl</code>
<code>\l__postnotes_prev_mark_chap_tl</code>	
<code>\l__postnotes_prev_mark_sect_tl</code>	
<code>\l__postnotes_prev_mark_name_tl</code>	


```

724 \tl_new:N \l__postnotes_prev_mark_page_tl
725 \tl_new:N \l__postnotes_prev_mark_chap_tl
726 \tl_new:N \l__postnotes_prev_mark_sect_tl
727 \tl_new:N \l__postnotes_prev_mark_name_tl

```

(End 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>
728 \cs_new_protected:Npn \__postnotes_get_headers_data:N #1
729 {
730   \group_begin:
731   \tl_gclear:N \pnhdpagefirst
732   \tl_gclear:N \pnhdpagelast
733   \tl_gclear:N \pnhdchapfirst
734   \tl_gclear:N \pnhdchaplast
735   \tl_gclear:N \pnhdsectfirst
736   \tl_gclear:N \pnhdsectlast
737   \tl_gclear:N \pnhdnamefirst
738   \tl_gclear:N \pnhdnamelast
739   \prop_gclear:N \g__postnotes_header_page_first_prop
740   \prop_gclear:N \g__postnotes_header_page_last_prop
741   \prop_gclear:N \g__postnotes_header_chap_first_prop
742   \prop_gclear:N \g__postnotes_header_chap_last_prop
743   \prop_gclear:N \g__postnotes_header_sect_first_prop
744   \prop_gclear:N \g__postnotes_header_sect_last_prop
745   \prop_gclear:N \g__postnotes_header_name_first_prop
746   \prop_gclear:N \g__postnotes_header_name_last_prop
747   \tl_gclear:N \g__postnotes_header_prev_last_page_tl
748   \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
749   \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
750   \tl_gclear:N \g__postnotes_header_prev_last_name_tl
751   \tl_clear:N \l__postnotes_prev_text_page_tl
752   \tl_clear:N \l__postnotes_curr_text_page_tl
753   \tl_clear:N \l__postnotes_prev_mark_page_tl
754   \tl_clear:N \l__postnotes_prev_mark_chap_tl
755   \tl_clear:N \l__postnotes_prev_mark_sect_tl
756   \tl_clear:N \l__postnotes_prev_mark_name_tl
757   \seq_map_inline:Nn #1
758   {
759     \exp_args:Nx \tl_if_eq:nnT
760     { \__postnotes_prop_item:nn {##1} { type } }
761     { note }
762     {
763       \__postnotes_get_pageref:Nn
764       \l__postnotes_curr_text_page_tl { text@ ##1 }
765       \tl_if_empty:NF \l__postnotes_curr_text_page_tl
766       {
767         \tl_if_eq:NNTF
768         \l__postnotes_prev_text_page_tl
769         \l__postnotes_curr_text_page_tl
770         {

```

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

```

771         \_postnotes_get_pageref:Nn
772         \l__postnotes_prev_mark_page_tl { mark@ ##1 }
773         \_postnotes_prop_get:nnN {##1} { thechapter }
774         \l__postnotes_prev_mark_chap_tl
775         \_postnotes_prop_get:nnN {##1} { thesection }
776         \l__postnotes_prev_mark_sect_tl
777         \_postnotes_prop_get:nnN {##1} { pnsectname }
778         \l__postnotes_prev_mark_name_tl
779     }
780 {

```

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).

```

781         \tl_if_empty:NF \l__postnotes_prev_text_page_tl
782         {
783             \prop_gput:Nxx \g__postnotes_header_page_last_prop
784             { \l__postnotes_prev_text_page_tl }
785             { \l__postnotes_prev_mark_page_tl }
786             \prop_gput:Nxx \g__postnotes_header_chap_last_prop
787             { \l__postnotes_prev_text_page_tl }
788             { \l__postnotes_prev_mark_chap_tl }
789             \prop_gput:Nxx \g__postnotes_header_sect_last_prop
790             { \l__postnotes_prev_text_page_tl }
791             { \l__postnotes_prev_mark_sect_tl }
792             \prop_gput:Nxx \g__postnotes_header_name_last_prop
793             { \l__postnotes_prev_text_page_tl }
794             { \l__postnotes_prev_mark_name_tl }
795         }

```

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

```

796         \prop_gput:Nxx \g__postnotes_header_page_first_prop
797         { \l__postnotes_curr_text_page_tl }
798         { \_postnotes_extract_pageref:n { mark@ ##1 } }
799         \prop_gput:Nxx \g__postnotes_header_chap_first_prop
800         { \l__postnotes_curr_text_page_tl }
801         { \_postnotes_prop_item:nn {##1} { thechapter } }
802         \prop_gput:Nxx \g__postnotes_header_sect_first_prop
803         { \l__postnotes_curr_text_page_tl }
804         { \_postnotes_prop_item:nn {##1} { thesection } }
805         \prop_gput:Nxx \g__postnotes_header_name_first_prop
806         { \l__postnotes_curr_text_page_tl }
807         { \_postnotes_prop_item:nn {##1} { pnsectname } }

```

Store `prev_mark` data for the first note on the page.

```

808         \_postnotes_get_pageref:Nn
809         \l__postnotes_prev_mark_page_tl { mark@ ##1 }
810         \_postnotes_prop_get:nnN {##1} { thechapter }
811         \l__postnotes_prev_mark_chap_tl

```

```

812         \l__postnotes_prop_get:nnN {##1} { thesection }
813         \l__postnotes_prev_mark_sect_tl
814         \l__postnotes_prop_get:nnN {##1} { pnsectname }
815         \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).
816         \tl_set:NV \l__postnotes_prev_text_page_tl
817         \l__postnotes_curr_text_page_tl
818     }
819 }
820 }
821 }

```

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.

```

822 \tl_if_empty:NF \l__postnotes_prev_text_page_tl
823 {
824   \prop_gput:Nxx \g__postnotes_header_page_last_prop
825   { \l__postnotes_prev_text_page_tl }
826   { \l__postnotes_prev_mark_page_tl }
827   \prop_gput:Nxx \g__postnotes_header_chap_last_prop
828   { \l__postnotes_prev_text_page_tl }
829   { \l__postnotes_prev_mark_chap_tl }
830   \prop_gput:Nxx \g__postnotes_header_sect_last_prop
831   { \l__postnotes_prev_text_page_tl }
832   { \l__postnotes_prev_mark_sect_tl }
833   \prop_gput:Nxx \g__postnotes_header_name_last_prop
834   { \l__postnotes_prev_text_page_tl }
835   { \l__postnotes_prev_mark_name_tl }
836 }
837 \group_end:
838 }

```

(End 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)}
839 \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
840 {
841   \group_begin:

```

```

842 \prop_get:NnNTF \g__postnotes_header_page_first_prop
843   {#1} \l_tmpa_tl
844   { \tl_gset:NV \pnhdpagefirst \l_tmpa_tl }
845   { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
846 \prop_get:NnNTF \g__postnotes_header_page_last_prop
847   {#1} \l_tmpa_tl
848   {
849     \tl_gset:NV \pnhdpagelast \l_tmpa_tl
850     \tl_gset:NV \g__postnotes_header_prev_last_page_tl \l_tmpa_tl
851   }
852   { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
853 \prop_get:NnNTF \g__postnotes_header_chap_first_prop
854   {#1} \l_tmpa_tl
855   { \tl_gset:NV \pnhdchapfirst \l_tmpa_tl }
856   { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
857 \prop_get:NnNTF \g__postnotes_header_chap_last_prop
858   {#1} \l_tmpa_tl
859   {
860     \tl_gset:NV \pnhdchaplast \l_tmpa_tl
861     \tl_gset:NV \g__postnotes_header_prev_last_chap_tl \l_tmpa_tl
862   }
863   { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
864 \prop_get:NnNTF \g__postnotes_header_sect_first_prop
865   {#1} \l_tmpa_tl
866   { \tl_gset:NV \pnhdsectfirst \l_tmpa_tl }
867   { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
868 \prop_get:NnNTF \g__postnotes_header_sect_last_prop
869   {#1} \l_tmpa_tl
870   {
871     \tl_gset:NV \pnhdsectlast \l_tmpa_tl
872     \tl_gset:NV \g__postnotes_header_prev_last_sect_tl \l_tmpa_tl
873   }
874   { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
875 \prop_get:NnNTF \g__postnotes_header_name_first_prop
876   {#1} \l_tmpa_tl
877   { \tl_gset:NV \pnhdnamefirst \l_tmpa_tl }
878   { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
879 \prop_get:NnNTF \g__postnotes_header_name_last_prop
880   {#1} \l_tmpa_tl
881   {
882     \tl_gset:NV \pnhdnamelast \l_tmpa_tl
883     \tl_gset:NV \g__postnotes_header_prev_last_name_tl \l_tmpa_tl
884   }
885   { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
886 \group_end:
887 }
888 \cs_generate_variant:Nn \__postnotes_set_headers_vars:n { x }

```

(End 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.

```

889 \AddToHook { shipout/before } [ postnotes/header ]
890   { \__postnotes_set_headers_vars_next: }
891 \bool_new:N \g__postnotes_header_vars_next_bool
892 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:
893   {
894     \bool_if:NT \g__postnotes_header_vars_next_bool
895       { \__postnotes_set_headers_vars:x { \int_eval:n { \c@page + 1 } } }
896   }
897 \cs_new_protected:Npn \__postnotes_set_headers_vars_first:
898   {
899     \__postnotes_set_print_page_label:x
900     { \int_use:N \g__postnotes_print_postnotes_int }
901     \__postnotes_set_headers_vars:x
902     {
903       \__postnotes_extract_pageref:e
904       { print@ \int_use:N \g__postnotes_print_postnotes_int }
905     }
906   }

```

(End 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
907 \NewDocumentCommand \pnheaderdefault {}
908   {
909     \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
910     { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
911     { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
912   }

```

(End definition for `\pnheaderdefault`.)

9 Compatibility

A dedicated temp variable for restoring data.

```

913 \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.

```

914 \AddToHook { postnotes/note/begin } [ postnotes ]
915 {
916   \cs_if_exist:NT \@capttype
917     { \bool_set_true:N \l__postnotes_maybe_multi_bool }
918 }

```

hyperref

```

919 \bool_new:N \g__postnotes_hyperref_loaded_bool
920 \AddToHook { package/hyperref/after }
921 { \bool_gset_true:N \g__postnotes_hyperref_loaded_bool }

```

```

\__postnotes_hyperref_make_currentHref:n
\__postnotes_hyperref_set_anchor:n
\__postnotes_ref_star:n

```

Auxiliary functions for hyperref support.

```

\__postnotes_hyperref_make_currentHref:n {<anchor/destination>}
\__postnotes_hyperref_set_anchor:n {<anchor/destination>}
\__postnotes_ref_star:n {<label>}

922 \cs_new_protected:Npn \__postnotes_hyperref_make_currentHref:n #1
923 {
924   \bool_if:NT \g__postnotes_hyperref_loaded_bool
925     { \Hy@MakeCurrentHref {#1} }
926 }
927 \cs_new_protected:Npn \__postnotes_hyperref_set_anchor:n #1
928 {
929   \bool_if:NT \g__postnotes_hyperref_loaded_bool
930     { \Hy@raisedlink { \hyper@anchor {#1} } }
931 }
932 \cs_new_protected:Npn \__postnotes_ref_star:n #1
933 {
934   \bool_if:NTF \g__postnotes_hyperref_loaded_bool
935     { \ref*{#1} }
936     { \ref{#1} }
937 }

```

(End definition for `__postnotes_hyperref_make_currentHref:n`, `__postnotes_hyperref_set_anchor:n`, and `__postnotes_ref_star:n`.)

biblatex

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

```

938 \AddToHook { package/biblatex/after }
939 {

```

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

```

940   \AddToHook { postnotes/print/begin } [ postnotes ]
941   { \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.

```

942   \AddToHook { begindocument/before } [ postnotes/mkbibendnote ]
943   {
944     \cs_set:Npn \blx@theendnote { \postnote }
945     \cs_set:Npn \blx@theendnotetext
946       { \blx@err@endnote \footnotetext }
947   }
948 }
949 (*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):

```

950 \AddToHook { package/biblatex/after }
951 {

```

Store `biblatex` variables for each note.

```

952   \AddToHook { postnotes/store/note } [ postnotes ]
953   {
954     \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
955     { biblatex@refsection } { \int_use:N \c@refsection }
956     \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
957     { biblatex@refsegment } { \int_use:N \c@refsegment }
958     \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
959     { biblatex@refcontextbool }
960     { \iftoggle { blx@refcontext } { true } { false } }
961     \prop_gput:cnV { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
962     { biblatex@refcontext } \blx@refcontext@context
963   }

```

`biblatex` setup, once for `\printpostnotes` call.

```

964   \AddToHook { postnotes/print/begin } [ postnotes ]
965   {
966     \__postnotes_biblatex_endrefcontext_local:
967     \__postnotes_biblatex_citereset_local:
968   }

```

Restore `biblatex` variables for each note.

```

969   \AddToHook { postnotes/print/eachnote } [ postnotes ]
970   {

```

```

971     \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
972     { biblatex@refsection } \l__postnotes_restore_tmp_tl
973     \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
974     \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
975     { biblatex@refsegment } \l__postnotes_restore_tmp_tl
976     \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }
977     \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
978     { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
979     \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
980     \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
981     { biblatex@refcontext } \l__postnotes_restore_tmp_tl
982     \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
983   }

```

Auxiliary functions.

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

```

984     \cs_new_protected:Npn \__postnotes_biblatex_endrefcontext_local:
985     {
986       \togglefalse { blx@refcontext }
987       \tl_clear:N \blx@refcontext@labelprefix
988       \tl_clear:N \blx@refcontext@labelprefix@real
989       \tl_set:Nx \blx@refcontext@sortingtemplatename { \blx@sorting }
990       \tl_set:Nn \blx@refcontext@sortingnamekeytemplatename { global }
991       \tl_set:Nn \blx@refcontext@uniquenametemplatename { global }
992       \tl_set:Nn \blx@refcontext@labelalphanametemplatename { global }
993       \blx@edef@refcontext
994       {
995         \blx@refcontext@sortingtemplatename /
996         \blx@refcontext@sortingnamekeytemplatename /
997         /
998         \blx@refcontext@uniquenametemplatename /
999         \blx@refcontext@labelalphanametemplatename
1000       }
1001     }

```

(End 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`.

```

1002     \cs_new_protected:Npn \__postnotes_biblatex_citereset_local:
1003     {
1004       \global\cslet{blx@bsee@{the\c@refsection}}\empty
1005       \global\cslet{blx@fsee@{the\c@refsection}}\empty
1006       \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1007       \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
1008       \blx@ibidreset@force
1009       \undef \blx@lastkey@text
1010       \undef \blx@lastkey@foot
1011       \blx@idemreset@force
1012       \undef \blx@lasthash@text
1013       \undef \blx@lasthash@foot

```



```

\blx@opcitreset@force
1010     \clist_map_inline:Nn \blx@trackhash@text
1011     { \csundef { blx@lastkey@text@ ##1 } }
1012     \tl_clear:N \blx@trackhash@text
1013     \clist_map_inline:Nn \blx@trackhash@foot
1014     { \csundef { blx@lastkey@foot@ ##1 } }
1015     \tl_clear:N \blx@trackhash@foot

\blx@loccitreset@force
1016     \clist_map_inline:Nn \blx@trackkeys@text
1017     { \csundef { blx@lastnote@text@ ##1 } }
1018     \tl_clear:N \blx@trackkeys@text
1019     \clist_map_inline:Nn \blx@trackkeys@foot
1020     { \csundef { blx@lastnote@foot@ ##1 } }
1021     \tl_clear:N \blx@trackkeys@foot

and all of them do:
1022     \cs_set_eq:NN \blx@lastmpfn \z@
1023     }

(End definition for \__postnotes_biblatex_citereset_local:.)

1024 }

```

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`.

```

1025 \AddToHook { package/biblatex/after }
1026 {
1027     \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1028     \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1029     \AddToHook { postnotes/print/begin } [ postnotes ]
1030     {
1031         \tl_set:Nx \l__postnotes_biblatex_orig_refsection_tl
1032         { \int_use:N \c@refsection }
1033         \tl_gset:Nx \g__postnotes_biblatex_prev_refsection_tl
1034         \l__postnotes_biblatex_orig_refsection_tl
1035     }
1036     \AddToHook { postnotes/print/eachnote } [ postnotes ]
1037     {
1038         \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }

```

```

1039     { biblatex@refsection } \l__postnotes_restore_tmp_tl
1040 \tl_if_eq:NNF
1041   \l__postnotes_restore_tmp_tl
1042   \g__postnotes_biblatex_prev_refsection_tl
1043   {
1044     \int_set:Nn \c@blx@maxsection
1045       { \l__postnotes_restore_tmp_tl - 1 }
1046     \tl_gset_eq:NN \g__postnotes_biblatex_prev_refsection_tl
1047       \l__postnotes_restore_tmp_tl
1048     \group_begin:
1049     \cs_set_eq:NN \label \use_none:n
1050     \cs_set_eq:NN \blx@info \use_none:n
1051     \blx@endrefsection
1052     \refsection
1053     \group_end:
1054   }
1055 }
1056 }
1057 </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:`.

```

1058 \tl_new:N \l__postnotes_note_zlabel_tl

```

(End definition for `\l__postnotes_note_zlabel_tl`.)

```

1059 \AddToHook { package/zref-user/after }
1060 {

```

Provide `zlabel` option.

```

1061   \keys_define:nn { postnotes/note }
1062   {
1063     zlabel .tl_set:N = \l__postnotes_note_zlabel_tl ,
1064     zlabel .value_required:n = true ,
1065   }

```

`\postnotezref` Provide `\postnotezref`.

```

\postnotezref(*){<label>}

```

```

1066   \NewDocumentCommand \postnotezref { s m }
1067   { \__postnotes_note_zref:nn {#1} {#2} }

```

(End definition for `\postnotezref`.)

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

```

\__postnotes_note_zref:nn {<star bool>} {<label>}

```

```

1068 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1069 {
1070   \group_begin:
1071   \__postnotes_typeset_mark_wrapper:n
1072   {
1073     \bool_lazy_and:nnTF
1074     { ! #1 }
1075     { \l__postnotes_hyperlink_bool }
1076     {
1077       \hyperlink
1078       { \zref@extractdefault {#2} { anchor } { } }
1079       { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1080     }
1081     { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1082   }
1083   \group_end:
1084 }

```

(End definition for __postnotes_note_zref:nn.)

```

1085 }

```

zref-clever

```

1086 \AddToHook { package/zref-clever/after }
1087 {
1088   \zcsetup
1089   {
1090     countertype = { postnote = endnote } ,
1091     countertype = { postnotetext = endnote } ,
1092   }
1093   \AddToHook { postnotes/print/begin } [ postnotes ]
1094   { \zcsetup { counterresetby = { postnotetext = postnotesection } } }
1095 }

```

zref-check

```

1096 \AddToHook { package/zref-check/after }
1097 {
1098   \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1099   {
1100     \AddToHook { postnotes/store/note } [ postnotes ]
1101     {
1102       \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1103       { zref-check@abschap } { \int_use:N \c@zc@abschap }
1104       \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1105       { zref-check@abssec } { \int_use:N \c@zc@abssec }
1106     }
1107     \AddToHook { postnotes/print/eachnote } [ postnotes ]
1108     {
1109       \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1110       { zref-check@abschap } \l__postnotes_restore_tmp_tl
1111       \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1112       \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1113       { zref-check@abssec } \l__postnotes_restore_tmp_tl

```

```

1114         \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }
1115     }
1116 }
1117 { }
1118 }

```

amsmath

```

1119 \AddToHook { package/amsmath/after }
1120 {

```

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

```

1121     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1122     {
1123         \legacy_if:nT { measuring@ }
1124         {
1125             \bool_set_true:N \l__postnotes_inhibit_note_bool
1126             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1127         }
1128     }

```

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.

```

1129     \AddToHook { postnotes/note/begin } [ postnotes ]
1130     {
1131         \legacy_if:nF { firstchoice@ }
1132         { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1133     }
1134 }

```

csquotes

```

1135 \AddToHook { package/csquotes/after }
1136 {
1137     \bool_new:N \l__postnotes_csquotes_measuring_bool
1138     \BlockquoteDisable
1139     { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1140     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1141     {
1142         \bool_if:NT \l__postnotes_csquotes_measuring_bool
1143         {
1144             \bool_set_true:N \l__postnotes_inhibit_note_bool
1145             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1146         }
1147     }
1148 }

```

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>.

```
1149 \AddToHook { package/tabularx/after }
1150 {
1151   \bool_new:N \l__postnotes_tabularx_inside_env_bool
1152   \AddToHook { env/tabularx/begin } [ postnotes ]
1153   {
1154     \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1155     \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1156   }
1157   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1158   {
1159     \bool_lazy_and:nnT
1160     { \l__postnotes_tabularx_inside_env_bool }
1161     { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1162     {
1163       \bool_set_true:N \l__postnotes_inhibit_note_bool
1164       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1165     }
1166   }
1167 }
```

tabularray

I've tried, but I could not find any “handle” to distinguish in `tabularray` a trial/measure pass from the final one. So we use `__postnotes_verify_multipass:N` for it.

```
1168 \AddToHook { package/tabularray/after }
1169 {
1170   \clist_map_inline:nn
1171   { tblr , longtblr , talltblr , booktabs , longtabs , talltabs , +array }
1172   {
1173     \AddToHook { env/#1/begin } [ postnotes ]
1174     { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1175   }
1176 }
```

10 Languages

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

```
1177 \tl_new:N \pntitle
1178 \tl_new:N \pnhdnotes
1179 \tl_new:N \pnhdtopage
1180 \tl_new:N \pnhdtopages
1181 \tl_set:Nn \pntitle { Notes }
1182 \tl_set:Nn \pnhdnotes { Notes }
1183 \tl_set:Nn \pnhdtopage { to~page }
1184 \tl_set:Nn \pnhdtopages { to~pages }
```

(End 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)}
1185 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
1186 {
1187   \tl_new:c { g__postnotes_language_ #1 _tl }
1188   \tl_gset:cn { g__postnotes_language_ #1 _tl } {#2}
1189 }
```

(End 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)}
1190 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
1191 {
1192   \ActivateGenericHook { babel/#1/beforeextras }
1193   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
1194     { g__postnotes_language_ #2 _tl }
1195 }
```

(End 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:nn` 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⟩}

1196 \cs_new_protected:Npn \__postnotes_set_polyglossia_language:nn #1#2
1197   {
1198     \AddToHook { package/polyglossia/after }
1199     {
1200       \exp_args:Nnv \csgappto { captions #1 }
1201       { g__postnotes_language_ #2 _tl }
1202     }
1203   }

```

(End definition for `__postnotes_set_polyglossia_language:nn`.)

English

```

1204 \__postnotes_define_language:nn { english }
1205   {
1206     \tl_set:Nn \pntitle      { Notes }
1207     \tl_set:Nn \pnhdnotes   { Notes }
1208     \tl_set:Nn \pnhdtopage  { to~page }
1209     \tl_set:Nn \pnhdtopages { to~pages }
1210   }
1211 \__postnotes_set_babel_language:nn { english } { english }
1212 \__postnotes_set_babel_language:nn { british } { english }
1213 \__postnotes_set_babel_language:nn { american } { english }
1214 \__postnotes_set_babel_language:nn { canadian } { english }
1215 \__postnotes_set_babel_language:nn { australian } { english }
1216 \__postnotes_set_babel_language:nn { newzealand } { english }
1217 \__postnotes_set_babel_language:nn { UKenglish } { english }
1218 \__postnotes_set_babel_language:nn { USenglish } { english }
1219 \__postnotes_set_polyglossia_language:nn { english } { english }

```

Portuguese

```

1220 \__postnotes_define_language:nn { portuguese }
1221   {
1222     \tl_set:Nn \pntitle      { Notas }
1223     \tl_set:Nn \pnhdnotes   { Notas }
1224     \tl_set:Nn \pnhdtopage  { da~página }
1225     \tl_set:Nn \pnhdtopages { das~páginas }
1226   }
1227 \__postnotes_set_babel_language:nn { portuguese } { portuguese }

```

```

1228 \_postnotes_set_babel_language:nn { brazilian } { portuguese }
1229 \_postnotes_set_babel_language:nn { portuges } { portuguese }
1230 \_postnotes_set_babel_language:nn { brazil } { portuguese }
1231 \_postnotes_set_polyglossia_language:nn { portuguese } { portuguese }

```

French

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

```

1232 \_postnotes_define_language:nn { french }
1233 {
1234   \tl_set:Nn \pntitle { Notes }
1235   \tl_set:Nn \pnhdnotes { Notes }
1236   \tl_set:Nn \pnhdtopage { de~la~page }
1237   \tl_set:Nn \pnhdtopages { des~pages }
1238 }
1239 \_postnotes_set_babel_language:nn { french } { french }
1240 \_postnotes_set_babel_language:nn { acadian } { french }
1241 \_postnotes_set_babel_language:nn { canadien } { french }
1242 \_postnotes_set_babel_language:nn { francais } { french }
1243 \_postnotes_set_babel_language:nn { frenchb } { french }
1244 \_postnotes_set_polyglossia_language:nn { french } { french }

```

German

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

```

1245 \_postnotes_define_language:nn { german }
1246 {
1247   \tl_set:Nn \pntitle { Endnoten }
1248   \tl_set:Nn \pnhdnotes { Endnoten }
1249   \tl_set:Nn \pnhdtopage { zu~Seite }
1250   \tl_set:Nn \pnhdtopages { zu~Seiten }
1251 }
1252 \_postnotes_set_babel_language:nn { german } { german }
1253 \_postnotes_set_babel_language:nn { ngerman } { german }
1254 \_postnotes_set_babel_language:nn { austrian } { german }
1255 \_postnotes_set_babel_language:nn { naustrian } { german }
1256 \_postnotes_set_babel_language:nn { swissgerman } { german }
1257 \_postnotes_set_babel_language:nn { nswissgerman } { german }
1258 \_postnotes_set_polyglossia_language:nn { german } { german }
1259 </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	\AddToHook	239 ,
		889, 914, 920, 938, 940, 942, 950,
\ActivateGenericHook	1192	952, 964, 969, 1025, 1029, 1036,
\addto	38	1059, 1086, 1093, 1096, 1100, 1107,
\addtocounter	36	1119, 1121, 1129, 1135, 1140, 1149,

1152, 1157, 1168, 1173, 1193, 1198	<code>\cs_set_eq:NN</code>	161, 178, 514, 1022, 1049, 1050, 1155
<code>\AddToHookNext</code>	<code>\csgappto</code>	1200
	<code>\csundef</code>	1011, 1014, 1017, 1020
B		
<code>\begin</code>		566
<code>\BlockquoteDisable</code>		1138
bool commands:		
<code>\bool_gset_false:N</code>		624
<code>\bool_gset_true:N</code>		509, 921
<code>\bool_if:NTF</code> ...		30, 244, 334, 336, 379, 392, 400, 507, 512, 565, 615, 634, 637, 650, 894, 924, 929, 934, 1142
<code>\bool_lazy_and:nnTF</code>		439, 686, 1073, 1159
<code>\bool_new:N</code>		135, 212, 213, 214, 266, 351, 354, 355, 371, 372, 891, 919, 1137, 1151
<code>\bool_set_false:N</code>		142, 221, 230, 231, 246, 376, 377
<code>\bool_set_true:N</code>		147, 220, 225, 226, 365, 917, 1125, 1126, 1132, 1139, 1144, 1145, 1154, 1163, 1164, 1174
<code>\bool_to_str:N</code>		46
<code>\bool_until_do:nn</code>		517
box commands:		
<code>\box_use:N</code>		291
<code>\box_wd:N</code>		290
<code>\l_tmpa_box</code>		289, 290, 291
C		
<code>\caption</code>		10, 11, 29
<code>\chapter</code>		116
<code>\citereset</code>		32
clist commands:		
<code>\clist_map_inline:Nn</code>		1010, 1013, 1016, 1019
<code>\clist_map_inline:nn</code>		1170
<code>\counterwithin</code>		12
cs commands:		
<code>\cs_generate_variant:Nn</code>		18, 80, 86, 92, 99, 106, 410, 466, 647, 888
<code>\cs_if_eq_p:NN</code>		1161
<code>\cs_if_exist:NTF</code>		34, 54, 95, 102, 112, 666, 916
<code>\cs_new:Npn</code>		16, 68, 100, 188
<code>\cs_new_protected:Npn</code>		20, 50, 63, 70, 75, 81, 87, 93, 114, 121, 313, 396, 412, 425, 434, 455, 495, 632, 648, 657, 675, 728, 839, 892, 897, 922, 927, 932, 984, 1002, 1068, 1185, 1190, 1196
<code>\cs_new_protected:Npx</code>		73
<code>\cs_set:Npn</code>		328, 583, 944, 945
<code>\cs_set:Npx</code>		329, 586
E		
<code>\end</code>		616
<code>\endgroup</code>		20
<code>\endinput</code>		12
<code>\endlist</code>		170, 187
<code>\endnotemark</code>		10
<code>\endnotetext</code>		10
<code>\endrefcontext</code>		32
<code>\enoteheading</code>		18
exp commands:		
<code>\exp_args:NNNx</code>		385
<code>\exp_args:Nnv</code>		1193, 1200
<code>\exp_args:NV</code>		428, 430
<code>\exp_args:Nx</code>		566, 616, 759
<code>\exp_not:N</code>		74
<code>\exp_not:n</code>		103
F		
<code>\fmtversion</code>		3
<code>\footnotemark</code>		9–11
<code>\footnotesize</code>		281
<code>\footnotetext</code>		9, 10, 946
fp commands:		
<code>\fp_compare:nNnTF</code>		692
<code>\fp_new:N</code>		352
<code>\fp_set:Nn</code>		364
<code>\fp_use:N</code>		31
G		
<code>\gappto</code>		38
group commands:		
<code>\group_begin:</code>		315, 383, 436, 457, 497, 540, 567, 570, 659, 677, 730, 841, 1048, 1070
<code>\group_end:</code>		348, 386, 448, 464, 557, 600, 617, 628, 673, 700, 837, 886, 1053, 1083
H		
<code>\hbox</code>		195
hbox commands:		
<code>\hbox_set:Nn</code>		289
<code>\hspace</code>		189
<code>\hyperlink</code>		1077
<code>\hyperref</code>		443
I		
<code>\IfFormatAtLeastTF</code>		3, 4
<code>\IfPackageAtLeastTF</code>		1098
<code>\IfPackageLoadedTF</code>		241

<code>\l__postnotes_clear_queue_seq</code> . . .	<code>__postnotes_hyperref_set_-</code>
. 21, 480, 505, 625	anchor:n 30, 338, 402, 636, 922, 927
<code>\l__postnotes_csquotes_measuring_-</code>	<code>\l__postnotes_hyperref_warn_bool</code>
bool 1137, 1139, 1142 213, 221, 226, 231, 244
<code>\l__postnotes_curr_text_page_tl</code> .	<code>__postnotes_inhibit_note:</code> 374
. 27, 710, 752,	<code>__postnotes_inhibit_note:TF</code> . . .
764, 765, 769, 797, 800, 803, 806, 817 22, 317, 371
<code>__postnotes_data_name:n</code> . . 2, 12,	<code>\l__postnotes_inhibit_note_bool</code> .
16, 16, 18, 22, 23, 24, 26, 28, 36, 39,	. . 14, 371, 376, 392, 1125, 1144, 1163
41, 43, 45, 47, 52, 53, 56, 59, 61, 65,	<code>__postnotes_list_makelabel:n</code> . . .
69, 71, 954, 956, 958, 961, 1102, 1104 161, 178, 188
<code>__postnotes_define_language:nn</code> .	<code>__postnotes_make_mark:nnn</code> . 192,
. 38,	390, 403, 407, 444, 446, 1079, 1081
39, 1185, 1185, 1204, 1220, 1232, 1245	<code>__postnotes_make_text_mark:nnn</code> .
<code>__postnotes_extract_pageref:n</code> 196, 639, 643, 645
. 4, 93, 100, 106, 798, 903	<code>\l__postnotes_manual_sortnum_-</code>
<code>__postnotes_get_headers_data:N</code> .	bool 30, 354, 365
. 23, 25, 27, 510, 728, 728	<code>\l__postnotes_mark_tl</code> . . 25, 320,
<code>__postnotes_get_pageref:Nn</code>	323, 329, 344, 350, 358, 381, 387, 390
. . . . 4, 93, 93, 99, 572, 763, 771, 808	<code>\l__postnotes_maybe_multi_bool</code> . .
<code>\g__postnotes_header_chap_first_-</code> 22, 36, 46, 355, 917, 1132, 1174
prop 710, 741, 799, 853	<code>\l__postnotes_nomark_bool</code>
<code>\g__postnotes_header_chap_last_-</code> 334, 351, 360
prop 710, 742, 786, 827, 857	<code>__postnotes_note:nn</code>
<code>\g__postnotes_header_name_first_-</code> 12, 14, 15, 311, 312, 313
prop 710, 745, 805, 875	<code>\g__postnotes_note_id_int</code>
<code>\g__postnotes_header_name_last_-</code> 12, 306, 319, 459
prop 710, 746, 792, 833, 879	<code>\l__postnotes_note_id_tl</code> 12,
<code>\g__postnotes_header_page_first_-</code>	306, 326, 331, 332, 339, 344, 346,
prop 710, 739, 796, 842	460, 463, 954, 956, 958, 961, 1102, 1104
<code>\g__postnotes_header_page_last_-</code>	<code>\l__postnotes_note_label_tl</code>
prop 710, 740, 783, 824, 846 353, 368, 427, 428
<code>\g__postnotes_header_prev_last_-</code>	<code>__postnotes_note_ref:nn</code>
chap_tl 710, 748, 856, 861, 863 15, 433, 434, 434
<code>\g__postnotes_header_prev_last_-</code>	<code>\l__postnotes_note_zlabel_tl</code>
name_tl 710, 750, 878, 883, 885 34, 429, 430, 1058, 1063
<code>\g__postnotes_header_prev_last_-</code>	<code>__postnotes_note_zref:nn</code>
page_tl 710, 747, 845, 850, 852 34, 1067, 1068, 1068
<code>\g__postnotes_header_prev_last_-</code>	<code>\l__postnotes_post_printnote_tl</code> .
sect_tl 710, 749, 867, 872, 874 143, 202, 209, 599
<code>\g__postnotes_header_sect_first_-</code>	<code>\l__postnotes_post_textmark_tl</code> . .
prop 710, 743, 802, 864 201, 207, 653, 655
<code>\g__postnotes_header_sect_last_-</code>	<code>\l__postnotes_pre_textmark_tl</code> . . .
prop 710, 744, 789, 830, 868 200, 205, 653, 655
<code>\g__postnotes_header_vars_next_-</code>	<code>\l__postnotes_prev_mark_chap_tl</code> .
bool 28, 509, 624, 891, 894 710, 754, 774, 788, 811, 829
<code>\l__postnotes_hyperlink_bool</code> . . .	<code>\l__postnotes_prev_mark_name_tl</code> .
. 212, 220, 710, 756, 778, 794, 815, 835
225, 230, 246, 336, 400, 441, 634, 1075	<code>\l__postnotes_prev_mark_page_tl</code> .
<code>\g__postnotes_hyperref_loaded_-</code> 710, 753, 772, 785, 809, 826
bool 919, 921, 924, 929, 934	<code>\l__postnotes_prev_mark_sect_tl</code> .
<code>__postnotes_hyperref_make_-</code> 710, 755, 776, 791, 813, 832
currentHref:n 30, 330, 588, 922, 922	

`\l__postnotes_prev_text_page_tl` .
 26, 27, 710, 751, 768, 781, 784, 787,
 790, 793, 816, 822, 825, 828, 831, 834
`\l__postnotes_print_as_list_bool`
 135, 142, 147, 512, 565, 615, 650
`\l__postnotes_print_content_tl` . .
 480, 555, 556, 582, 598
`\l__postnotes_print_counter_tl` . .
 480, 579, 585
`\l__postnotes_print_env_tl`
 134, 144, 148, 566, 616
`\l__postnotes_print_format_tl`
 127, 130, 568
`\l__postnotes_print_mark_tl`
 480, 576, 587, 596
`\l__postnotes_print_note_id_-`
 next_tl 480,
 527, 532, 534, 548, 551, 603, 608, 610
`\l__postnotes_print_note_id_tl`
 480, 520, 521, 542, 545, 554,
 573, 575, 578, 581, 589, 593, 595,
 971, 974, 977, 980, 1038, 1109, 1112
`__postnotes_print_notes:`
 17, 18, 21, 23, 474, 495, 495
`\l__postnotes_print_plain_mark_-`
 bool
 14, 372, 377, 379, 1126, 1145, 1164
`\g__postnotes_print_postnotes_-`
 int 480, 498, 900, 904
`\l__postnotes_print_type_curr_tl`
 480, 522, 523, 559, 620
`\l__postnotes_print_type_next_tl`
 480, 528, 535, 537, 604, 611, 613
`\l__postnotes_print_type_prev_tl`
 480, 504, 558, 563, 619
`__postnotes_prop_gclear:n`
 3, 63, 70, 626
`__postnotes_prop_get:nnN`
 3, 63, 63, 521, 533, 541,
 544, 547, 550, 553, 574, 577, 580,
 609, 663, 680, 681, 684, 685, 690,
 691, 773, 775, 777, 810, 812, 814,
 971, 974, 977, 980, 1038, 1109, 1112
`__postnotes_prop_item:nn`
 3, 63, 68, 760, 801, 804, 807
`\g__postnotes_queue_seq` 12, 22,
 23, 25, 306, 325, 460, 499, 505, 506,
 508, 510, 517, 519, 525, 531, 601, 607
`\c__postnotes_ref_prefix_tl`
 4, 72, 74, 95, 96, 102, 103, 667
`__postnotes_ref_star:n`
 30, 446, 922, 932
`\l__postnotes_restore_tmp_tl`
 913, 972, 973, 975,
 976, 978, 979, 981, 982, 1039, 1041,
 1045, 1047, 1110, 1111, 1113, 1114
`\l__postnotes_saved_spacefactor_-`
 tl 411, 417, 422
`\g__postnotes_sectid_int` 44, 454, 458
`__postnotes_section:nn`
 16, 451, 453, 454, 455, 466
`\g__postnotes_section_name_tl`
 42, 461, 467, 470
`__postnotes_set_babel_language:nn`
 38, 1190,
 1190, 1211, 1212, 1213, 1214, 1215,
 1216, 1217, 1218, 1227, 1228, 1229,
 1230, 1239, 1240, 1241, 1242, 1243,
 1252, 1253, 1254, 1255, 1256, 1257
`__postnotes_set_headers_vars:n`
 25, 27, 28, 839, 839, 888, 895, 901
`__postnotes_set_headers_vars_-`
 first: 24, 29, 511, 889, 897
`__postnotes_set_headers_vars_-`
 next: 24, 28, 889, 890, 892
`__postnotes_set_mark_page_-`
 label:n 4, 23, 75, 75, 80, 332
`__postnotes_set_polyglossia_-`
 language:nn 39,
 1196, 1196, 1219, 1231, 1244, 1258
`__postnotes_set_print_page_-`
 label:n 4, 23, 75, 87, 92, 899
`__postnotes_set_text_page_-`
 label:n 4, 23, 75, 81, 86, 592
`__postnotes_set_user_labels:`
 34, 333, 425, 425
`\l__postnotes_sort_bool` 266, 269, 507
`\l__postnotes_sort_num_fp` 31, 352, 364
`__postnotes_sort_queue:N`
 23, 508, 675, 675
`__postnotes_store:nn` 2, 19, 20, 346
`__postnotes_store_section:nn`
 3, 50, 50, 463
`\l__postnotes_tabularx_inside_-`
 env_bool 1151, 1154, 1160
`__postnotes_tabularx_saved_-`
 write:Nn 1155, 1161
`__postnotes_text_mark_wrapper:n`
 21, 590, 632, 648
`__postnotes_typeset_mark:nn`
 14, 343, 396, 396, 410
`__postnotes_typeset_mark_-`
 wrapper:n
 14, 389, 396, 398, 412, 437, 1071
`__postnotes_typeset_text_-`
 mark:nn 21, 594, 632, 632, 647
`__postnotes_verify_multipass:N`
 22, 37, 506, 657, 657

<code>\blx@trackhash@text</code>	1010, 1012	<code>\tl_if_empty:NTF</code>	320, 381, 427, 429, 765, 781, 822
<code>\blx@trackkeys@foot</code>	1019, 1021	<code>\tl_if_eq:NNTF</code>	682, 767, 909, 1040
<code>\blx@trackkeys@text</code>	1016, 1018	<code>\tl_if_eq:NnTF</code>	523, 537, 563, 613, 664
<code>\c@blx@maxsection</code>	1044	<code>\tl_if_eq:nnTF</code>	140, 759
<code>\c@page</code>	84, 90, 895	<code>\tl_new:N</code>	127, 134, 200, 201, 202, 307, 350, 353, 411, 467, 475, 476, 477, 478, 479, 481, 482, 483, 484, 485, 486, 487, 488, 702, 703, 704, 705, 706, 707, 708, 709, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 913, 1027, 1028, 1058, 1177, 1178, 1179, 1180, 1187
<code>\c@postnote</code>	14, 27, 32, 384	<code>\tl_set:Nn</code>	96, 143, 144, 148, 308, 323, 387, 417, 504, 527, 528, 558, 603, 604, 619, 816, 989, 990, 991, 992, 1031, 1181, 1182, 1183, 1184, 1206, 1207, 1208, 1209, 1222, 1223, 1224, 1225, 1234, 1235, 1236, 1237, 1247, 1248, 1249, 1250
<code>\c@postnotetext</code>	584	<code>\l_tmpa_tl</code>	663, 664, 680, 682, 684, 687, 690, 692, 843, 844, 847, 849, 850, 854, 855, 858, 860, 861, 865, 866, 869, 871, 872, 876, 877, 880, 882, 883
<code>\c@refsection</code>	955, 973, 1004, 1005, 1032	<code>\l_tmpb_tl</code>	681, 682, 685, 688, 691, 692
<code>\c@refsegment</code>	957, 976	<code>\togglefalse</code>	986
<code>\c@z@abschap</code>	1103, 1111	<code>\toggletrue</code>	941
<code>\c@z@abssec</code>	1105, 1114	token commands:	
<code>\Hy@MakeCurrentHref</code>	925	<code>\token_to_str:N</code>	78, 84, 90
<code>\Hy@raisedlink</code>	930	<code>\topsep</code>	166, 183
<code>\hyper@anchor</code>	930		
<code>\hyper@linkend</code>	405, 641		
<code>\hyper@linkstart</code>	404, 640		
<code>\ifmeasuring@</code>	36		
<code>\p@postnote</code>	329, 587		
<code>\post@note</code>	4, 72, 78, 84, 90		
<code>\z@</code>	1022		
<code>\zref@extractdefault</code>	1078		
<code>\text</code>	10, 11, 36		
<code>\textnormal</code>	36		
<code>\textup</code>	36		
<code>\the</code>	417		
<code>\thechapter</code>	24, 37, 57		
<code>\theHpostnote</code>	12		
<code>\thepage</code>	23, 78		
<code>\theHpostnote</code>	14, 323, 387		
<code>\thesection</code>	24, 40, 60		
tl commands:			
<code>\c_empty_tl</code>	104		
<code>\tl_clear:N</code>	66, 97, 751, 752, 753, 754, 755, 756, 987, 988, 1004, 1005, 1012, 1015, 1018, 1021		
<code>\tl_const:Nn</code>	72		
<code>\tl_gclear:N</code>	461, 731, 732, 733, 734, 735, 736, 737, 738, 747, 748, 749, 750		
<code>\tl_gput_right:Nn</code>	38		
<code>\tl_gset:Nn</code>	844, 845, 849, 850, 852, 855, 856, 860, 861, 863, 866, 867, 871, 872, 874, 877, 878, 882, 883, 885, 1033, 1188		
<code>\tl_gset_eq:NN</code>	1046		
		U	
		<code>\undef</code>	1006, 1007, 1008, 1009
		use commands:	
		<code>\use:N</code>	979
		<code>\use_none:n</code>	1049, 1050
		<code>\UseHook</code>	48, 327, 378, 503, 571
		W	
		<code>\write</code>	1155, 1161
		Z	
		<code>\zcsetup</code>	1088, 1094
		<code>\zlabel</code>	430
		<code>\zref</code>	1079, 1081