

The `postnotes` package^{*}

Code documentation

Gustavo Barros[†]

2023-06-12

Contents

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

^{*}This file describes v0.2.4, released 2023-06-12.

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

1 Initial setup

Start the DocStrip guards.

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

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

```
3 \providecommand\IfFormatAtLeastTF{\@ifl@t@r\fmtversion}
4 \IfFormatAtLeastTF{2022-06-01}
5 {}
6 {%
7   \PackageError{postnotes}{LaTeX kernel too old}
8   {%
9     'postnotes' requires a LaTeX kernel 2022-06-01 or newer.%
10    \MessageBreak Loading will abort!%
11   }%
12 \endinput
13 }%
14 \ProvidesExplPackage {postnotes} {2023-06-12} {0.2.4}
15 {Endnotes for LaTeX}
```

2 Data

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

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

(End of definition for _postnotes_data_name:n.)

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

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

```
\_postnotes_store:nn {\<note id>} {\<note content>}
```

```

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

```

(End of definition for __postnotes_store:nn.)

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

```

\__postnotes_store_section:nn {\langle \text{note id} \rangle} {\langle \text{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 }
63 \cs_generate_variant:Nn \__postnotes_store_section:nn { nx }

```

(End of definition for __postnotes_store_section:nn.)

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

```

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

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

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

```

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

```

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

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

(End of definition for \post@note.)

```

`__postnotes_set_mark_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_text_page_label:n`
`__postnotes_set_print_page_label:n`

```

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

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

```

(End of 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}

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

```

(End of definition for `_postnotes_get_pageref:Nn` and `_postnotes_extract_pageref:n`.)

3 Options

heading option

```

108 \keys_define:nn { postnotes/setup }
109 {
110   heading .cs_set_protected:Np = \pnheading ,
111   heading .value_required:n = true ,
112 }

```

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

```

113 \cs_if_exist:cTF { chapter }
114 {
115   \cs_new_protected:Npn \pnheading
116   {
117     \chapter*{\pntitle}
118     \@mkboth{\pnheaderdefault}{\pnheaderdefault}
119   }
120 }
121 {
122   \cs_new_protected:Npn \pnheading
123   {
124     \section*{\pntitle}
125     \@mkboth{\pnheaderdefault}{\pnheaderdefault}
126   }
127 }

```

(End of definition for `\pnheading`.)

format option

```
128 \tl_new:N \l__postnotes_print_format_tl
129 \keys_define:nn { postnotes/setup }
130 {
131     format .tl_set:N = \l__postnotes_print_format_tl ,
132     format .initial:n = { \small } ,
133     format .value_required:n = true ,
134 }
```

listenv option

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

```

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

```
145         \tl_set:Nn \l__postnotes_print_env_tl { itemize }
146     }
147     {
148         \bool_set_true:N \l__postnotes_print_as_list_bool
149         \tl_set:Nn \l__postnotes_print_env_tl {#1}
150     }
151 },
152 listenv .initial:n = { postnoteslist } ,
153 listenv .value_required:n = true ,
154 }
```

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*).

```
155 \NewDocumentEnvironment { postnoteslist } { }
156 {
157     \list { }
158     {
159         \setlength { \leftmargin } { Opt }
160         \setlength { \labelwidth } { Opt }
161         \setlength { \itemindent } { .5\parindent }
162         \cs_set_eq:NN \makelabel \l__postnotes_list_makelabel:n
163         \setlength { \rightmargin } { Opt }
164         \setlength { \listparindent } { \parindent }
165         \setlength { \parsep } { \parskip }
166         \setlength { \itemsep } { Opt }
167         \setlength { \topsep } { .5\topsep }
168         \setlength { \partopsep } { .5\partopsep }
169     }
170 }
171 { \endlist }
172 \NewDocumentEnvironment { postnoteslisthang } { }
```

```

173 {
174   \list { }
175   {
176     \setlength { \leftmargin } { 1em }
177     \setlength { \labelwidth } { -\leftmargin }
178     \setlength { \itemindent } { -2\leftmargin }
179     \cs_set_eq:NN \makelabel \_postnotes_list_makelabel:n
180     \setlength { \rightmargin } { 0pt }
181     \setlength { \listparindent } { \parindent }
182     \setlength { \parsep } { \parskip }
183     \setlength { \itemsep } { 0pt }
184     \setlength { \topsep } { .5\topsep }
185     \setlength { \partopsep } { .5\partopsep }
186   }
187 }
188 { \endlist }
189 \cs_new:Npn \_postnotes_list_makelabel:n #1
190 { \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.

```

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

```

From the default kernel definition of \@makefnmark.

```

195   makemark .initial:n =
196     { #2 \hbox { \@textsuperscript { \normalfont #1 } } #3 } ,
197   maketextmark .cs_set:Np = \_postnotes_make_text_mark:nnn #1#2#3 ,
198   maketextmark .value_required:n = true ,
199   maketextmark .initial:n = { #2 #1 . #3 } ,
200 }

```

pretextmark, posttextmark, postprintnote options

```

201 \tl_new:N \l__postnotes_pre_textmark_tl
202 \tl_new:N \l__postnotes_post_textmark_tl
203 \tl_new:N \l__postnotes_post_printnote_tl
204 \keys_define:nn { postnotes/setup }
205 {
206   pretextmark .tl_set:N = \l__postnotes_pre_textmark_tl ,
207   pretextmark .value_required:n = true ,
208   posttextmark .tl_set:N = \l__postnotes_post_textmark_tl ,
209   posttextmark .value_required:n = true ,
210   postprintnote .tl_set:N = \l__postnotes_post_printnote_tl ,
211   postprintnote .value_required:n = true ,
212 }

```

hyperref and backlink options

```

213 \bool_new:N \l__postnotes_hyperlink_bool
214 \bool_new:N \l__postnotes_hyperref_warn_bool

```

```

215 \bool_new:N \l__postnotes_backlink_bool
216 \keys_define:nn { postnotes/setup }
217 {
218   hyperref .choice: ,
219   hyperref / auto .code:n =
220   {
221     \bool_set_true:N \l__postnotes_hyperlink_bool
222     \bool_set_false:N \l__postnotes_hyperref_warn_bool
223   } ,
224   hyperref / true .code:n =
225   {
226     \bool_set_true:N \l__postnotes_hyperlink_bool
227     \bool_set_true:N \l__postnotes_hyperref_warn_bool
228   } ,
229   hyperref / false .code:n =
230   {
231     \bool_set_false:N \l__postnotes_hyperlink_bool
232     \bool_set_false:N \l__postnotes_hyperref_warn_bool
233   } ,
234   hyperref .initial:n = auto ,
235   hyperref .default:n = true ,
236   backlink .bool_set:N = \l__postnotes_backlink_bool ,
237   backlink .initial:n = true ,
238   backlink .default:n = true ,
239 }
240 \AddToHook { begindocument }
241 {
242   \IfPackageLoadedTF { hyperref }
243   { }
244   {
245     \bool_if:NT \l__postnotes_hyperref_warn_bool
246     { \msg_warning:nn { postnotes } { missing-hyperref } }
247     \bool_set_false:N \l__postnotes_hyperlink_bool
248   }
249   \keys_define:nn { postnotes/setup }
250   {
251     hyperref .code:n =
252     {
253       \msg_warning:nnn { postnotes }
254       { option-preamble-only } { hyperref }
255     } ,
256     backlink .code:n =
257     {
258       \msg_warning:nnn { postnotes }
259       { option-preamble-only } { backlink }
260     } ,
261   }
262 }
263 \msg_new:nnn { postnotes } { option-preamble-only }
264 { Option~'#1'~only~available~in~the~preamble~\msg_line_context:. }
265 \msg_new:nnn { postnotes } { missing-hyperref }
266 { Missing~'hyperref'~package.~Setting~'hyperref=false'. }

```


sort option

```
267 \bool_new:N \l__postnotes_sort_bool
268 \keys_define:nn { postnotes/setup }
269 {
270   sort .bool_set:N = \l__postnotes_sort_bool ,
271   sort .initial:n = true ,
272   sort .default:n = true ,
273 }
```

style option

```
274 \keys_define:nn { postnotes/setup }
275 {
276   style .choice: ,
277   style / endnotes .meta:n =
278   {
279     listenv = none ,
280     format =
281     {
282       \footnotesize
283       \setlength { \rightskip } { 0pt }
284       \setlength { \leftskip } { 0pt }
285       \setlength { \parindent } { 1.8em }
286     } ,
287     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.

```
288     maketextmark =
289     {
290       \hbox_set:Nn \l_tmpa_box { \@textsuperscript { \normalfont ##1 } }
291       \skip_horizontal:n { - \box_wd:N \l_tmpa_box }
292       ##2 \box_use:N \l_tmpa_box ##3
293     } ,
294   } ,
295   style / pagenote .meta:n =
296   {
297     listenv = none ,
298     format = { } ,
299     pretextmark = { \par\noindent } ,
300     maketextmark = { { \normalfont ##2 ##1 . ##3 } } ,
301     posttextmark = { ~ } ,
302   } ,
303 }
```

\postnotesetup

\postnotesetup Provide \postnotesetup.

\postnotesetup{<options>}

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

(End of definition for \postnotesetup.)

4 `\postnote`

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

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

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

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

Multiple typesetting passes of content are much harder. And they abound: the standard classes’ `\caption` typesets the caption once, if it is short, but twice if it is longer than a line; `amsmath`’s math environments perform a measuring pass before actually typesetting the equations; `amsmath`’s `\text` macro runs the contents through `\mathchoice` (which typesets the contents four times) when in math mode; `tabularx` and `tabularray` also perform measuring passes of their tables; so does `csquotes`’ blockquotes; and certainly more that I’m unaware. A number of these places offer some one or another way to mitigate the issue: `amsmath`, `tabularx`, `csquotes` and (optionally) `tabularray` restore counter values after measuring steps; `amsmath` offers a boolean to indicate when it is a measuring pass; `csquotes` offers further handles. But the standard `\caption` offers none, and neither does `amsmath`’s `\text` macro. Well, the `pkgcaption` package can disable the multiple passes for `\caption` with the option `singlelinecheck`, but it is not reasonable to require it for our purposes, so we must assume the worst case.

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

In this scenario, when there’s no way around the multiple passes, `\footnotemark` can only handle the general case if used with an argument, precisely because it inhibits the stepping of the counter. Otherwise the counter is stepped multiple times, and we’d get the wrong number (and mark). In some circumstances, if we know the number of passes is deterministic, we might get away by adjusting the counter manually (`\caption` may be dealt with this way: if we know it to be two lines, we can decrement the counter before it and get correct results, even hyperlinked). But in cases which adjusting the counter is sufficient, end notes can be dealt with in the same way, and doesn’t need the

separation between “mark” and “text”. So, what is distinctive of the kernel’s footnote apparatus, which allows it as much flexibility as one would like, is receiving an arbitrary number as argument and not stepping the counter. And as far as `\footnotemark` and `\footnotetext` are concerned, the main point of the optional argument is really to “manually establish the relation” between the two of them. So, if *not stepping the counter* is what is needed to handle the general case, is it viable to do so? What would we lose in so doing?

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

I have not been able to devise a way to “reconnect” “text” and “mark” in the absence of the unique ID counter. The most promising idea was to have mandatory arguments to `\postnotemark` and `\postnotetext` receiving a *⟨label⟩* which we could use to identify their counterparts, but I was not able to go through with this, and the attempts all increased complexity considerably. It is not just a label/ref system, there’s got to be a one-to-one correspondence between the sets, uniqueness has to be ensured on both sides, and there cannot be “lone” marks or texts (a bijection). Besides, this label based system of identification would have to live side-by-side with the one based on the counter. So, even if we’d have unique IDs, we wouldn’t know beforehand in what form it comes. Considering the ID is used to build the variable name in which we store the note’s information, this would also complicate things.

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

The truth is there are some trade-offs, there’s no “ideal” solution. Still, all in all, my judgment is that the unique ID counter is worth more than the inconveniences of an occasional `\postnote[nomark]` referenced with `\postnoteref`, and even that should not

be needed much. So, for the time being, until something else shakes this balance, I won't be offering `\postnotemark` and `\postnotetext`.

For the `hyperref` support for cross-references in `\postnote`, I've moved back and forth quite a lot. One of the ideas I fancied was using `\refstepcounter` and let `hyperref` do its job. But, since I want to have control of the anchor/destination name on both "sides", I'd have to set `\theHpostnote` locally before calling `\refstepcounter`, otherwise results might 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.

```
306 \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 to be processed by the next call of `\printpostnotes`.

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

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

`\postnote` Provide `\postnote`.

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

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

(End of definition for `\postnote`.)

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

```
\__postnotes_note:nn [<options>] {\<note content>}
```

```

313 \NewHook { postnotes/note/begin }
314 \cs_new_protected:Npn \__postnotes_note:nn #1#2
315 {
316   \group_begin:
317   \keys_set:nn { postnotes/note } {#1}
318   \__postnotes_inhibit_note:F
319   {
320     \int_gincr:N \g__postnotes_note_id_int
321     \tl_if_empty:NT \l__postnotes_mark_tl
322     {
323       \stepcounter { postnote }
324       \tl_set:Nx \l__postnotes_mark_tl { \thepostnote }
325     }
326     \seq_gput_right:Nx \g__postnotes_queue_seq
327     { \l_postnotes_note_id_tl }
328     \UseHook { postnotes/note/begin }
329     \cs_set:Npn \@currentcounter { postnote }
330     \cs_set:Npx \@currentlabel { \p@postnote \l__postnotes_mark_tl }
331     \MakeLinkTarget* { 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:NF \l__postnotes_nomark_bool
335     {
336       \__postnotes_typeset_mark:xV
337       { \l_postnotes_note_id_tl } \l__postnotes_mark_tl
338     }
339     \__postnotes_store:nn { \l_postnotes_note_id_tl } {#2}
340   }
341   \group_end:
342 }

```

(End of definition for __postnotes_note:nn.)

Options for \postnote.

```

343 \tl_new:N \l__postnotes_mark_tl
344 \bool_new:N \l__postnotes_nomark_bool
345 \fp_new:N \l__postnotes_sort_num_fp
346 \tl_new:N \l__postnotes_note_label_tl
347 \bool_new:N \l__postnotes_manual_sortnum_bool
348 \bool_new:N \l__postnotes_maybe_multi_bool
349 \keys_define:nn { postnotes/note }
350 {
351   markstr .tl_set:N = \l__postnotes_mark_tl ,
352   markstr .value_required:n = true ,
353   sortnum .code:n =
354   {
355     \fp_set:Nn \l__postnotes_sort_num_fp {#1}
356     \bool_set_true:N \l__postnotes_manual_sortnum_bool
357   } ,
358   sortnum .value_required:n = true ,
359   mark .meta:n =
360   {
361     markstr = {#1} ,
362     sortnum = {#1} ,

```

```

363     } ,
364     mark .value_required:n = true ,
365     nomark .bool_set:N = \l__postnotes_nomark_bool ,
366     nomark .default:n = true ,
367     label .tl_set:N = \l__postnotes_note_label_tl ,
368     label .value_required:n = true ,
369 }

```

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

```

370 \bool_new:N \l__postnotes_inhibit_note_bool
371 \bool_new:N \l__postnotes_print_plain_mark_bool
372 \bool_new:N \l__postnotes_print_plain_mark_stepcounter_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     \bool_set_false:N \l__postnotes_print_plain_mark_stepcounter_bool
379     \UseHook { postnotes/note/inhibit }

```

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

```

380     \bool_if:NT \l__postnotes_print_plain_mark_bool
381     {
382         \tl_if_empty:NT \l__postnotes_mark_tl
383         {
384             \bool_if:NTF \l__postnotes_print_plain_mark_stepcounter_bool
385             {
386                 \stepcounter { postnote }
387                 \tl_set:Nx \l__postnotes_mark_tl { \thepostnote }
388             }
389             {
390                 \group_begin:
391                 \int_incr:N \c@postnote
392                 \exp_args:NNNx
393                 \group_end:
394                 \tl_set:Nn \l__postnotes_mark_tl { \thepostnote }
395             }
396         }

```

```

397     \_postnotes_typeset_mark_wrapper:n
398     { \_postnotes_make_mark:nnn { \l__postnotes_mark_tl } { } { } }
399   }
400   \bool_if:NTF \l__postnotes_inhibit_note_bool
401   { \prg_return_true:  }
402   { \prg_return_false: }
403 }

```

(End of definition for _postnotes_inhibit_note:TF.)

_postnotes_typeset_mark:nn Auxiliary functions for mark typesetting in _postnotes_note:nn. _postnotes_typeset_mark_wrapper:n is based on the definition of \@footnotemark in the kernel.

```

    \_postnotes_typeset_mark:nn {<note id>} {<mark>}
    \_postnotes_typeset_mark_wrapper:n {<mark>}

404 \cs_new_protected:Npn \_postnotes_typeset_mark:nn #1#2
405 {
406   \_postnotes_typeset_mark_wrapper:n
407   {
408     \bool_if:NTF \l__postnotes_hyperlink_bool
409     {
410       \_postnotes_make_mark:nnn {#2}
411       { \hyper@linkstart { link } { postnote. #1 .text } }
412       { \hyper@linkend }
413     }
414     { \_postnotes_make_mark:nnn {#2} { } { } }
415   }
416 }
417 \cs_generate_variant:Nn \_postnotes_typeset_mark:nn { xV }
418 \tl_new:N \l__postnotes_saved_spacefactor_tl
419 \cs_new_protected:Npn \_postnotes_typeset_mark_wrapper:n #1
420 {
421   \mode_leave_vertical:
422   \mode_if_horizontal:T
423   {
424     \tl_set:Nx \l__postnotes_saved_spacefactor_tl { \the\spacefactor }
425     \nobreak
426   }
427   #1
428   \mode_if_horizontal:T
429   { \spacefactor \l__postnotes_saved_spacefactor_tl }
430   \scan_stop:
431 }

```

(End of 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.

```

432 \cs_new_protected:Npn \_postnotes_set_user_labels:
433 {
434   \tl_if_empty:NF \l__postnotes_note_label_tl
435   { \exp_args:NV \label \l__postnotes_note_label_tl }
436   \tl_if_empty:NF \l__postnotes_note_zlabel_tl

```

```

437     { \exp_args:NV \zlabel \l__postnotes_note_zlabel_tl }
438   }

```

(End of definition for `__postnotes_set_user_labels:.`)

5 `\postnoteref`

`\postnoteref` Provide `\postnoteref`.

```

\postnoteref{<*>}{<label>}

439 \NewDocumentCommand \postnoteref { s m }
440   { \__postnotes_note_ref:nn {#1} {#2} }

```

(End of definition for `\postnoteref`.)

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

```

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

441 \cs_new_protected:Npn \__postnotes_note_ref:nn #1#2
442   {
443     \group_begin:
444     \__postnotes_typeset_mark_wrapper:n
445       {
446         \bool_lazy_and:nnTF
447           { ! #1 }
448           { \l__postnotes_hyperlink_bool }
449           {
450             \hyperref [ #2 ]
451               { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
452           }
453           { \__postnotes_make_mark:nnn { \ref*{#2} } { } { } }
454       }
455     \group_end:
456   }

```

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

6 `\postnotessection`

`\postnotessection` Provide `\postnotessection` and `\postnotessectionx`.
`\postnotessectionx`

```

\postnotessection[<options>]{<section content>}
\postnotessectionx[<options>]{<section content>}

457 \NewDocumentCommand \postnotessection { 0 { } +m }
458   { \__postnotes_section:nn {#1} {#2} }
459 \NewDocumentCommand \postnotessectionx { 0 { } +m }
460   {
461     % NOTE Command deprecated in 2022-12-27 for v0.2.0.
462     \msg_warning:nn { postnotes } { postnotessectionx-deprecated }
463     \postnotessection [ #1 , exp ] {#2}

```



```

464 }
465 \msg_new:nnn { postnotes } { postnotessectionx-deprecated }
466 {
467   '\iow_char:N\postnotessectionx'~is~deprecated~\msg_line_context:~
468   Use~the~'exp'~option~of~'\iow_char:N\postnotessection'~instead.
469 }

```

(End of definition for \postnotessection and \postnotessectionx.)

__postnotes_section:nn The internal version of \postnotessection.

```

      \__postnotes_section:nn {<options>} {<content>}
470 \int_new:N \g__postnotes_sectid_int
471 \cs_new_protected:Npn \__postnotes_section:nn #1#2
472 {
473   \group_begin:
474   \int_gincr:N \g__postnotes_sectid_int
475   \int_gincr:N \g__postnotes_note_id_int
476   \seq_gput_right:Nx \g__postnotes_queue_seq { \l_postnotes_note_id_tl }
477   \tl_gclear:N \g__postnotes_section_name_tl
478   \keys_set:nn { postnotes/section } {#1}
479   \bool_if:NTF \l__postnotes_section_exp_bool
480     { \__postnotes_store_section:nx { \l_postnotes_note_id_tl } {#2} }
481     { \__postnotes_store_section:nn { \l_postnotes_note_id_tl } {#2} }
482   \group_end:
483 }

```

(End of 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.

```

484 \tl_new:N \g__postnotes_section_name_tl
485 \bool_new:N \l__postnotes_section_exp_bool
486 \keys_define:nn { postnotes/section }
487 {
488   name .tl_gset:N = \g__postnotes_section_name_tl ,
489   name .value_required:n = true ,
490   exp .bool_set:N = \l__postnotes_section_exp_bool ,
491   exp .initial:n = false ,
492   exp .default:n = true ,
493 }

```

7 \printpostnotes

\printpostnotes Provide \printpostnotes.

```

      \printpostnotes
494 \NewDocumentCommand \printpostnotes { }
495 { \__postnotes_print_notes: }

```

(End of definition for \printpostnotes.)

<code>\pnthechapter</code>	User facing variables, aimed at making available some of the notes' and sections' metadata
<code>\pnthesection</code>	for the user at specific contexts.
<code>\pnthechapternextnote</code>	496 <code>\tl_new:N \pnthechapter</code>
<code>\pnthesectionnextnote</code>	497 <code>\tl_new:N \pnthesection</code>
<code>\pnthepage</code>	498 <code>\tl_new:N \pnthechapternextnote</code>
	499 <code>\tl_new:N \pnthesectionnextnote</code>
	500 <code>\tl_new:N \pnthepage</code>

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

<code>\g_postnotes_print_postnotes_int</code>	Auxiliary variables for <code>__postnotes_print_notes:</code>
<code>\l_postnotes_print_note_id_tl</code>	501 <code>\int_new:N \g_postnotes_print_postnotes_int</code>
<code>\l__postnotes_print_note_id_next_tl</code>	502 <code>\tl_new:N \l_postnotes_print_note_id_tl</code>
<code>\l__postnotes_print_counter_tl</code>	503 <code>\tl_new:N \l__postnotes_print_note_id_next_tl</code>
<code>\l__postnotes_print_mark_tl</code>	504 <code>\tl_new:N \l__postnotes_print_counter_tl</code>
<code>\l_postnotes_print_type_curr_tl</code>	505 <code>\tl_new:N \l__postnotes_print_mark_tl</code>
<code>\l_postnotes_print_type_next_tl</code>	506 <code>\tl_new:N \l_postnotes_print_type_curr_tl</code>
<code>\l_postnotes_print_type_prev_tl</code>	507 <code>\tl_new:N \l__postnotes_print_type_next_tl</code>
<code>\l__postnotes_print_content_tl</code>	508 <code>\tl_new:N \l__postnotes_print_type_prev_tl</code>
<code>\l__postnotes_clear_queue_seq</code>	509 <code>\tl_new:N \l__postnotes_print_content_tl</code>
	510 <code>\seq_new:N \l__postnotes_clear_queue_seq</code>

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

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

```

511 \NewHook { postnotes/print/begin }
512 \NewHook { postnotes/print/note/begin }

```

The `postnotetext` is a counter used to restore the original value of `postnote` at the time of printing, for the purposes of cross-referencing, it should be different from `postnote` if a note may occur inside `\printpostnotes`. The `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 (larger than the conceivable number of postnote sections in a document), so that "marks" and "texts" don't mix in the same reference list, which would occur if the enclosing counters of both belonged to the same range, and with somewhat arbitrary results, since we cannot ensure the step of the enclosing counter along the document matches `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.

```

513 \newcounter { postnotetext }
514 \newcounter { postnotessection }
515 \setcounter { postnotessection } { 10000 }

```

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

```

\__postnotes_print_notes:

516 \cs_new_protected:Npn \__postnotes_print_notes:
517 {
518   \group_begin:
519   \int_gincr:N \g__postnotes_print_postnotes_int
520   \seq_if_empty:NTF \g__postnotes_queue_seq
521     { \msg_warning:nn { postnotes } { empty-printpostnotes } }
522     {
523       \pnheading
524       \UseHook { postnotes/print/begin }
525       \tl_set:Nn \l__postnotes_print_type_prev_tl { open }
526       \seq_set_eq:NN \l__postnotes_clear_queue_seq \g__postnotes_queue_seq
527       \__postnotes_verify_multipass:N \g__postnotes_queue_seq
528       \bool_if:NT \l__postnotes_sort_bool
529         { \__postnotes_sort_queue:N \g__postnotes_queue_seq }
530       \bool_gset_true:N \g__postnotes_header_vars_next_bool
531       \__postnotes_get_headers_data:N \g__postnotes_queue_seq
532       \__postnotes_set_headers_vars_first:

```

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

```

533   \bool_if:NF \l__postnotes_print_as_list_bool
534   {
535     \cs_set_eq:NN \@afterindentfalse \@afterindenttrue
536     \@afterindenttrue
537   }
538   \bool_until_do:nn { \seq_if_empty_p:N \g__postnotes_queue_seq }
539   {
540     \seq_gpop_left:NN \g__postnotes_queue_seq
541     \l_postnotes_print_note_id_tl
542     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
543       { type } \l__postnotes_print_type_curr_tl
544     \tl_if_eq:NnTF \l__postnotes_print_type_curr_tl { section }
545       { % type_curr = 'section'
546         \seq_if_empty:NTF \g__postnotes_queue_seq
547         {
548           \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
549           \tl_set:Nn \l__postnotes_print_type_next_tl { close }
550         }
551         {
552           \seq_get_left:NN \g__postnotes_queue_seq
553           \l__postnotes_print_note_id_next_tl

```

```

554         \__postnotes_prop_get:nnN
555         { \l__postnotes_print_note_id_next_tl }
556         { type } \l__postnotes_print_type_next_tl
557     }

```

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

```

558     \tl_if_eq:NnT \l__postnotes_print_type_next_tl { note }
559     {
560         \stepcounter { postnotessection }
561         \group_begin:
562         \__postnotes_prop_get:nnN
563         { \l_postnotes_print_note_id_tl }
564         { thechapter } \pnthechapter
565         \__postnotes_prop_get:nnN
566         { \l_postnotes_print_note_id_tl }
567         { thesection } \pnthesection
568         \__postnotes_prop_get:nnN
569         { \l__postnotes_print_note_id_next_tl }
570         { thechapter } \pnthechapternextnote
571         \__postnotes_prop_get:nnN
572         { \l__postnotes_print_note_id_next_tl }
573         { thesection } \pnthesectionnextnote
574         \__postnotes_prop_get:nnN
575         { \l_postnotes_print_note_id_tl }
576         { content } \l__postnotes_print_content_tl
577         \l__postnotes_print_content_tl
578         \group_end:

```

Set type_prev for the next iteration.

```

579         \tl_set:NV \l__postnotes_print_type_prev_tl
580         \l__postnotes_print_type_curr_tl
581     }
582 }
583 { % type_curr = 'note'
584     \tl_if_eq:NnF \l__postnotes_print_type_prev_tl { note }
585     {
586         \bool_if:NTF \l__postnotes_print_as_list_bool
587         { \exp_args:Nx \begin { \l__postnotes_print_env_tl } }
588         { \group_begin: }
589         \l__postnotes_print_format_tl
590     }
591     \group_begin:
592     \UseHook { postnotes/print/note/begin }
593     \__postnotes_get_pageref:Nx \pnthechapter
594     { mark@ \l_postnotes_print_note_id_tl }
595     \__postnotes_prop_get:nnN
596     { \l_postnotes_print_note_id_tl }
597     { mark } \l__postnotes_print_mark_tl
598     \__postnotes_prop_get:nnN
599     { \l_postnotes_print_note_id_tl }
600     { counter } \l__postnotes_print_counter_tl
601     \__postnotes_prop_get:nnN
602     { \l_postnotes_print_note_id_tl }
603     { content } \l__postnotes_print_content_tl
604     \cs_set:Npn \@currentcounter { postnotetext }

```

```

605 \int_set:Nn \c@postnotetext
606 { \l__postnotes_print_counter_tl }
607 \cs_set:Npx \@currentlabel
608 { \p@postnote \l__postnotes_print_mark_tl }
609 \__postnotes_text_mark_wrapper:n
610 {
611   \MakeLinkTarget*
612   { postnote. \l__postnotes_print_note_id_tl .text }
613   \__postnotes_set_text_page_label:x
614   { \l__postnotes_print_note_id_tl }
615   \__postnotes_typeset_text_mark:eV
616   { \l__postnotes_print_note_id_tl }
617   \l__postnotes_print_mark_tl
618 }
619 \l__postnotes_print_content_tl
620 \l__postnotes_post_printnote_tl
621 \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.

```

622 \seq_if_empty:NTF \g__postnotes_queue_seq
623 {
624   \tl_set:Nn \l__postnotes_print_note_id_next_tl { noid }
625   \tl_set:Nn \l__postnotes_print_type_next_tl { close }
626 }
627 {
628   \seq_get_left:NN \g__postnotes_queue_seq
629   \l__postnotes_print_note_id_next_tl
630   \__postnotes_prop_get:nnN
631   { \l__postnotes_print_note_id_next_tl }
632   { type } \l__postnotes_print_type_next_tl
633 }
634 \tl_if_eq:NnF \l__postnotes_print_type_next_tl { note }
635 {
636   \bool_if:NTF \l__postnotes_print_as_list_bool
637   { \exp_args:Nx \end { \l__postnotes_print_env_tl } }
638   { \group_end: }
639 }

```

Set `type_prev` for the next iteration.

```

640 \tl_set:NV \l__postnotes_print_type_prev_tl
641 \l__postnotes_print_type_curr_tl
642 }
643 }
644 \AddToHookNext { shipout/after }
645 { \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.

```

646 \seq_map_inline:Nn \l__postnotes_clear_queue_seq
647 { \__postnotes_prop_gclear:n { ##1 } }
648 }

```

```

649     \group_end:
650 }

```

(End of definition for `__postnotes_print_notes:`)

```

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

```

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

```

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

653 \cs_new_protected:Npn \__postnotes_typeset_text_mark:nn #1#2
654 {
655     \bool_lazy_and:nnTF
656     { \l__postnotes_hyperlink_bool }
657     { \l__postnotes_backlink_bool }
658     {
659         \__postnotes_make_text_mark:nnn {#2}
660         { \hyper@linkstart { link } { postnote. #1 .mark } }
661         { \hyper@linkend }
662     }
663     { \__postnotes_make_text_mark:nnn {#2} { } { } }
664 }
665 \cs_generate_variant:Nn \__postnotes_typeset_text_mark:nn { eV }
666 \cs_new_protected:Npn \__postnotes_text_mark_wrapper:n #1
667 {
668     \bool_if:NTF \l__postnotes_print_as_list_bool
669     {
670         \item [ \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl ]

```

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

```

671     \mode_leave_vertical:
672 }
673 { \l__postnotes_pre_textmark_tl #1 \l__postnotes_post_textmark_tl }
674 }

```

(End of definition for `__postnotes_typeset_text_mark:nn` 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 it’s 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
675 \cs_new_protected:Npn \__postnotes_verify_multipass:N #1
676 {
677   \group_begin:
678   \seq_clear:N \l_tmpa_seq
679   \seq_map_inline:Nn #1
680   {
681     \__postnotes_prop_get:nnN {##1} { multibool } \l_tmpa_tl
682     \tl_if_eq:NnTF \l_tmpa_tl { true }
683     {
684       \cs_if_exist:cT
685       { \c__postnotes_ref_prefix_tl @ mark@ ##1 }
686       { \seq_put_right:Nn \l_tmpa_seq {##1} }
687     }
688     { \seq_put_right:Nn \l_tmpa_seq {##1} }
689   }
690   \seq_gset_eq:NN #1 \l_tmpa_seq
691   \group_end:
692 }
```

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

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

```
\__postnotes_sort_queue:N \g__postnotes_queue_seq
693 \cs_new_protected:Npn \__postnotes_sort_queue:N #1
694 {
695   \group_begin:
696   \seq_gsort:Nn #1
697   {
698     \__postnotes_prop_get:nnN {##1} { pnsectid } \l_tmpa_tl
699     \__postnotes_prop_get:nnN {##2} { pnsectid } \l_tmpb_tl
700     \tl_if_eq:NNTF \l_tmpa_tl \l_tmpb_tl
701     {
702       \__postnotes_prop_get:nnN {##1} { type } \l_tmpa_tl
703       \__postnotes_prop_get:nnN {##2} { type } \l_tmpb_tl
704       \bool_lazy_and:nnTF
705       { \str_if_eq_p:Vn \l_tmpa_tl { note } }
706       { \str_if_eq_p:Vn \l_tmpb_tl { note } }
707       {
708         \__postnotes_prop_get:nnN {##1} { sortnum } \l_tmpa_tl
709         \__postnotes_prop_get:nnN {##2} { sortnum } \l_tmpb_tl
710         \fp_compare:nNnTF { \l_tmpa_tl } > { \l_tmpb_tl }
711         { \sort_return_swapped: }
712         { \sort_return_same: }
713       }
714     }
715     { \sort_return_same: }
```

```

715         }
716         { \sort_return_same: }
717     }
718     \group_end:
719 }

```

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

8 Headers

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

1. For each `\postnote`, labels are set storing the `page` where the note occurs. Each note actually generates a pair of such labels, once when `\postnote` is called (with the mark), and another where the note is printed (in `\printpostnotes`). The former ones store `\thepage`, since we want the printed representation of it for typesetting purposes, the latter ones store the value of the `page` counter, since we don't need to typeset it, but do need to perform algebraic operations with it. These labels are set by `__postnotes_set_mark_page_label:n`, `__postnotes_set_text_page_label:n`, and `__postnotes_set_print_page_label:n` at the appropriate places. The set of these labels provides a mapping from each note's "mark" and "text" to the page where it occurs.
2. This information set is processed by `__postnotes_get_headers_data:N` at the beginning of `__postnotes_print_notes:` to identify the first and last note of each page in `\printpostnotes`, and to generate a mapping from these first and last notes on each page to the pages where their corresponding marks occur. We also take the opportunity to enrich this mapping with other metadata of each note. So we get also mappings from the first and last note on each page to `\thechapter`, `\thesection`, and the `name` of the section in which they occur. These mappings are stored in property lists `\g__postnotes_header_{info}_first_prop` and `\g__postnotes_header_{info}_last_prop` where the `key` is the page in `\printpostnotes` where their note's content is typeset (or rather where it starts to be typeset, it is the page where the text's mark is printed).
3. Based on these mappings, along the span of notes section we run `__postnotes_set_headers_vars_next:` at each `shipout/before` hook to set user facing variables for the *next* page, which will be available when their heading gets typeset. Given that at `shipout` we can rely on a correct value of the `page` counter, we use it as `key` to query the property lists generated in the previous step. These user facing variables are called `\pnhd{info}first` and `\pnhd{info}last`. Since we cannot rely on the shipout hook for the first page of `\printpostnotes`, `__postnotes_set_headers_vars_first:` is run at its beginning to ensure correct values are in place on the first page of the notes section.

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

`\pnhdpagefirst` `\pnhdpagelast` `\pnhdchapfirst` `\pnhdchaplast` `\pnhdsectfirst` `\pnhdsectlast` `\pnhdnamefirst` `\pnhdnamelast` User facing variables, aimed at making available header data for the user. Setting these variables with correct values at the moment the header gets typeset is *the* objective of the whole headers infrastructure of the package.

```

720 \tl_new:N \pnhdpagefirst
721 \tl_new:N \pnhdpagelast
722 \tl_new:N \pnhdchapfirst
723 \tl_new:N \pnhdchaplast
724 \tl_new:N \pnhdsectfirst
725 \tl_new:N \pnhdsectlast
726 \tl_new:N \pnhdnamefirst
727 \tl_new:N \pnhdnamelast

```

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

`\g__postnotes_header_page_first_prop` `\g__postnotes_header_page_last_prop` `\g__postnotes_header_chap_first_prop` `\g__postnotes_header_chap_last_prop` `\g__postnotes_header_sect_first_prop` `\g__postnotes_header_sect_last_prop` `\g__postnotes_header_name_first_prop` `\g__postnotes_header_name_last_prop` `\g__postnotes_header_prev_last_page_tl` `\g__postnotes_header_prev_last_chap_tl` `\g__postnotes_header_prev_last_sect_tl` `\g__postnotes_header_prev_last_name_tl` `\l__postnotes_prev_text_page_tl` `\l__postnotes_curr_text_page_tl` `\l__postnotes_prev_mark_page_tl` `\l__postnotes_prev_mark_chap_tl` `\l__postnotes_prev_mark_sect_tl` `\l__postnotes_prev_mark_name_tl` Auxiliary variables for the headers infrastructure.

```

728 \prop_new:N \g__postnotes_header_page_first_prop
729 \prop_new:N \g__postnotes_header_page_last_prop
730 \prop_new:N \g__postnotes_header_chap_first_prop
731 \prop_new:N \g__postnotes_header_chap_last_prop
732 \prop_new:N \g__postnotes_header_sect_first_prop
733 \prop_new:N \g__postnotes_header_sect_last_prop
734 \prop_new:N \g__postnotes_header_name_first_prop
735 \prop_new:N \g__postnotes_header_name_last_prop
736 \tl_new:N \g__postnotes_header_prev_last_page_tl
737 \tl_new:N \g__postnotes_header_prev_last_chap_tl
738 \tl_new:N \g__postnotes_header_prev_last_sect_tl
739 \tl_new:N \g__postnotes_header_prev_last_name_tl
740 \tl_new:N \l__postnotes_prev_text_page_tl
741 \tl_new:N \l__postnotes_curr_text_page_tl
742 \tl_new:N \l__postnotes_prev_mark_page_tl
743 \tl_new:N \l__postnotes_prev_mark_chap_tl
744 \tl_new:N \l__postnotes_prev_mark_sect_tl
745 \tl_new:N \l__postnotes_prev_mark_name_tl

```

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

`__postnotes_get_headers_data:N` `__postnotes_set_headers_vars:n` Process header data for `__postnotes_set_headers_vars:n`.

```

\__postnotes_get_headers_data:N <\g__postnotes_queue_seq>
746 \cs_new_protected:Npn \__postnotes_get_headers_data:N #1
747 {
748   \group_begin:
749   \tl_gclear:N \pnhdpagefirst
750   \tl_gclear:N \pnhdpagelast
751   \tl_gclear:N \pnhdchapfirst
752   \tl_gclear:N \pnhdchaplast
753   \tl_gclear:N \pnhdsectfirst
754   \tl_gclear:N \pnhdsectlast
755   \tl_gclear:N \pnhdnamefirst
756   \tl_gclear:N \pnhdnamelast
757   \prop_gclear:N \g__postnotes_header_page_first_prop
758   \prop_gclear:N \g__postnotes_header_page_last_prop
759   \prop_gclear:N \g__postnotes_header_chap_first_prop

```

```

760 \prop_gclear:N \g__postnotes_header_chap_last_prop
761 \prop_gclear:N \g__postnotes_header_sect_first_prop
762 \prop_gclear:N \g__postnotes_header_sect_last_prop
763 \prop_gclear:N \g__postnotes_header_name_first_prop
764 \prop_gclear:N \g__postnotes_header_name_last_prop
765 \tl_gclear:N \g__postnotes_header_prev_last_page_tl
766 \tl_gclear:N \g__postnotes_header_prev_last_chap_tl
767 \tl_gclear:N \g__postnotes_header_prev_last_sect_tl
768 \tl_gclear:N \g__postnotes_header_prev_last_name_tl
769 \tl_clear:N \l__postnotes_prev_text_page_tl
770 \tl_clear:N \l__postnotes_curr_text_page_tl
771 \tl_clear:N \l__postnotes_prev_mark_page_tl
772 \tl_clear:N \l__postnotes_prev_mark_chap_tl
773 \tl_clear:N \l__postnotes_prev_mark_sect_tl
774 \tl_clear:N \l__postnotes_prev_mark_name_tl
775 \seq_map_inline:Nn #1
776 {
777   \exp_args:Nx \tl_if_eq:nnT
778   { \__postnotes_prop_item:nn {##1} { type } }
779   { note }
780   {
781     \__postnotes_get_pageref:Nn
782     \l__postnotes_curr_text_page_tl { text@ ##1 }
783     \tl_if_empty:NF \l__postnotes_curr_text_page_tl
784     {
785       \tl_if_eq:NNTF
786       \l__postnotes_prev_text_page_tl
787       \l__postnotes_curr_text_page_tl
788       {

```

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

```

789     \__postnotes_get_pageref:Nn
790     \l__postnotes_prev_mark_page_tl { mark@ ##1 }
791     \__postnotes_prop_get:nnN {##1} { thechapter }
792     \l__postnotes_prev_mark_chap_tl
793     \__postnotes_prop_get:nnN {##1} { thesection }
794     \l__postnotes_prev_mark_sect_tl
795     \__postnotes_prop_get:nnN {##1} { pnsectname }
796     \l__postnotes_prev_mark_name_tl
797   }
798 {

```

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).

```

799   \tl_if_empty:NF \l__postnotes_prev_text_page_tl
800   {
801     \prop_gput:Nxx \g__postnotes_header_page_last_prop
802     { \l__postnotes_prev_text_page_tl }
803     { \l__postnotes_prev_mark_page_tl }

```

```

804         \prop_gput:Nxx \g__postnotes_header_chap_last_prop
805         { \l__postnotes_prev_text_page_tl }
806         { \l__postnotes_prev_mark_chap_tl }
807     \prop_gput:Nxx \g__postnotes_header_sect_last_prop
808     { \l__postnotes_prev_text_page_tl }
809     { \l__postnotes_prev_mark_sect_tl }
810     \prop_gput:Nxx \g__postnotes_header_name_last_prop
811     { \l__postnotes_prev_text_page_tl }
812     { \l__postnotes_prev_mark_name_tl }
813 }

```

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

```

814     \prop_gput:Nxx \g__postnotes_header_page_first_prop
815     { \l__postnotes_curr_text_page_tl }
816     { \__postnotes_extract_pageref:n { mark@ ##1 } }
817     \prop_gput:Nxx \g__postnotes_header_chap_first_prop
818     { \l__postnotes_curr_text_page_tl }
819     { \__postnotes_prop_item:nn {##1} { thechapter } }
820     \prop_gput:Nxx \g__postnotes_header_sect_first_prop
821     { \l__postnotes_curr_text_page_tl }
822     { \__postnotes_prop_item:nn {##1} { thesection } }
823     \prop_gput:Nxx \g__postnotes_header_name_first_prop
824     { \l__postnotes_curr_text_page_tl }
825     { \__postnotes_prop_item:nn {##1} { pnsectname } }

```

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

```

826     \__postnotes_get_pageref:Nn
827     \l__postnotes_prev_mark_page_tl { mark@ ##1 }
828     \__postnotes_prop_get:nnN {##1} { thechapter }
829     \l__postnotes_prev_mark_chap_tl
830     \__postnotes_prop_get:nnN {##1} { thesection }
831     \l__postnotes_prev_mark_sect_tl
832     \__postnotes_prop_get:nnN {##1} { pnsectname }
833     \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).

```

834         \tl_set:NV \l__postnotes_prev_text_page_tl
835         \l__postnotes_curr_text_page_tl
836     }
837 }
838 }
839 }

```

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.

```

840     \tl_if_empty:NF \l__postnotes_prev_text_page_tl
841     {
842         \prop_gput:Nxx \g__postnotes_header_page_last_prop
843         { \l__postnotes_prev_text_page_tl }
844         { \l__postnotes_prev_mark_page_tl }
845         \prop_gput:Nxx \g__postnotes_header_chap_last_prop
846         { \l__postnotes_prev_text_page_tl }
847         { \l__postnotes_prev_mark_chap_tl }

```

```

848         \prop_gput:Nxx \g__postnotes_header_sect_last_prop
849         { \l__postnotes_prev_text_page_tl }
850         { \l__postnotes_prev_mark_sect_tl }
851         \prop_gput:Nxx \g__postnotes_header_name_last_prop
852         { \l__postnotes_prev_text_page_tl }
853         { \l__postnotes_prev_mark_name_tl }
854     }
855     \group_end:
856 }

```

(End of definition for __postnotes_get_headers_data:N.)

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

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

```

\__postnotes_set_headers_vars:n {(page number)}

857 \cs_new_protected:Npn \__postnotes_set_headers_vars:n #1
858 {
859     \group_begin:
860     \prop_get:NnNTF \g__postnotes_header_page_first_prop
861     {#1} \l_tmpa_tl
862     { \tl_gset:NV \pnhdpagefirst \l_tmpa_tl }
863     { \tl_gset:NV \pnhdpagefirst \g__postnotes_header_prev_last_page_tl }
864     \prop_get:NnNTF \g__postnotes_header_page_last_prop
865     {#1} \l_tmpa_tl
866     {
867         \tl_gset:NV \pnhdpagelast \l_tmpa_tl
868         \tl_gset:NV \g__postnotes_header_prev_last_page_tl \l_tmpa_tl
869     }
870     { \tl_gset:NV \pnhdpagelast \g__postnotes_header_prev_last_page_tl }
871     \prop_get:NnNTF \g__postnotes_header_chap_first_prop
872     {#1} \l_tmpa_tl
873     { \tl_gset:NV \pnhdchapfirst \l_tmpa_tl }
874     { \tl_gset:NV \pnhdchapfirst \g__postnotes_header_prev_last_chap_tl }
875     \prop_get:NnNTF \g__postnotes_header_chap_last_prop
876     {#1} \l_tmpa_tl
877     {
878         \tl_gset:NV \pnhdchaplast \l_tmpa_tl
879         \tl_gset:NV \g__postnotes_header_prev_last_chap_tl \l_tmpa_tl
880     }
881     { \tl_gset:NV \pnhdchaplast \g__postnotes_header_prev_last_chap_tl }
882     \prop_get:NnNTF \g__postnotes_header_sect_first_prop
883     {#1} \l_tmpa_tl
884     { \tl_gset:NV \pnhdsectfirst \l_tmpa_tl }

```

```

885     { \tl_gset:NV \pnhdsectfirst \g__postnotes_header_prev_last_sect_tl }
886 \prop_get:NnNTF \g__postnotes_header_sect_last_prop
887   {#1} \l_tmpa_tl
888   {
889     \tl_gset:NV \pnhdsectlast \l_tmpa_tl
890     \tl_gset:NV \g__postnotes_header_prev_last_sect_tl \l_tmpa_tl
891   }
892   { \tl_gset:NV \pnhdsectlast \g__postnotes_header_prev_last_sect_tl }
893 \prop_get:NnNTF \g__postnotes_header_name_first_prop
894   {#1} \l_tmpa_tl
895   { \tl_gset:NV \pnhdnamefirst \l_tmpa_tl }
896   { \tl_gset:NV \pnhdnamefirst \g__postnotes_header_prev_last_name_tl }
897 \prop_get:NnNTF \g__postnotes_header_name_last_prop
898   {#1} \l_tmpa_tl
899   {
900     \tl_gset:NV \pnhdnamelast \l_tmpa_tl
901     \tl_gset:NV \g__postnotes_header_prev_last_name_tl \l_tmpa_tl
902   }
903   { \tl_gset:NV \pnhdnamelast \g__postnotes_header_prev_last_name_tl }
904 \group_end:
905 }
906 \cs_generate_variant:Nn \__postnotes_set_headers_vars:n { x }

```

(End of definition for __postnotes_set_headers_vars:n.)

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

```

907 \AddToHook { shipout/before } [ postnotes/header ]
908   { \__postnotes_set_headers_vars_next: }
909 \bool_new:N \g__postnotes_header_vars_next_bool
910 \cs_new_protected:Npn \__postnotes_set_headers_vars_next:
911   {
912     \bool_if:NT \g__postnotes_header_vars_next_bool
913       { \__postnotes_set_headers_vars:x { \int_eval:n { \c@page + 1 } } }
914   }
915 \cs_new_protected:Npn \__postnotes_set_headers_vars_first:
916   {
917     \__postnotes_set_print_page_label:x
918     { \int_use:N \g__postnotes_print_postnotes_int }
919     \__postnotes_set_headers_vars:x
920     {
921       \__postnotes_extract_pageref:e
922       { print@ \int_use:N \g__postnotes_print_postnotes_int }
923     }
924   }

```

(End of definition for __postnotes_set_headers_vars_next: and __postnotes_set_headers_vars_first:.)

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

```

\pnheaderdefault

925 \NewDocumentCommand \pnheaderdefault {}
926 {
927   \tl_if_eq:NNTF \pnhdpagefirst \pnhdpagelast
928     { \pnhdnotes{} ~ \pnhdtopage{} ~ \pnhdpagefirst }
929     { \pnhdnotes{} ~ \pnhdtopages{} ~ \pnhdpagefirst -- \pnhdpagelast }
930 }

```

(End of definition for \pnheaderdefault.)

9 Compatibility

A dedicated temp variable for restoring data.

```

931 \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.

```

932 \AddToHook { postnotes/note/begin } [ postnotes ]
933 {
934   \cs_if_exist:NT \@captype
935     { \bool_set_true:N \l__postnotes_maybe_multi_bool }
936 }

```

biblatex

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

```

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

```

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

```

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

```

941   \AddToHook { begindocument/before } [ postnotes/mkbibendnote ]

```

```

942     {
943       \cs_set:Npn \blx@theendnote { \postnote }
944       \cs_set:Npn \blx@theendnotetext
945         { \blx@err@endnote \footnotetext }
946     }
947 }
948 <*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):

```

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

```

Store biblatex variables for each note.

```

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

```

biblatex setup, once for `\printpostnotes` call.

```

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

```

Restore biblatex variables for each note.

```

968   \AddToHook { postnotes/print/note/begin } [ postnotes ]
969   {
970     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
971       { biblatex@refsection } \l__postnotes_restore_tmp_tl
972     \int_set:Nn \c@refsection { \l__postnotes_restore_tmp_tl }
973     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
974       { biblatex@refsegment } \l__postnotes_restore_tmp_tl
975     \int_set:Nn \c@refsegment { \l__postnotes_restore_tmp_tl }

```

```

976     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
977     { biblatex@refcontextbool } \l__postnotes_restore_tmp_tl
978     \use:c { toggle \l__postnotes_restore_tmp_tl } { blx@refcontext }
979     \__postnotes_prop_get:nnN { \l_postnotes_print_note_id_tl }
980     { biblatex@refcontext } \l__postnotes_restore_tmp_tl
981     \blx@edef@refcontext { \l__postnotes_restore_tmp_tl }
982 }

```

Auxiliary functions.

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

```

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

```

(End of definition for `__postnotes_biblatex_endrefcontext_local:`)

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

```

1001     \cs_new_protected:Npn \__postnotes_biblatex_citereset_local:
1002     {
1003         \global\cslet{blx@bsee@the\c@refsection}\empty
1004         \global\cslet{blx@fsee@the\c@refsection}\empty
1005         \tl_clear:c { blx@bsee@ \int_use:N \c@refsection }
1006         \tl_clear:c { blx@fsee@ \int_use:N \c@refsection }
1007         \blx@ibidreset@force
1008         \undef \blx@lastkey@text
1009         \undef \blx@lastkey@foot
1010         \blx@idemreset@force
1011         \undef \blx@lasthash@text
1012         \undef \blx@lasthash@foot
1013         \blx@opcitereset@force
1014         \clist_map_inline:Nn \blx@trackhash@text
1015         { \csundef { blx@lastkey@text@ ##1 } }
1016         \tl_clear:N \blx@trackhash@text
1017         \clist_map_inline:Nn \blx@trackhash@foot

```



```

1013         { \csundef { blx@lastkey@foot@ ##1 } }
1014         \tl_clear:N \blx@trackhash@foot
\blx@loccitreset@force
1015         \clist_map_inline:Nn \blx@trackkeys@text
1016         { \csundef { blx@lastnote@text@ ##1 } }
1017         \tl_clear:N \blx@trackkeys@text
1018         \clist_map_inline:Nn \blx@trackkeys@foot
1019         { \csundef { blx@lastnote@foot@ ##1 } }
1020         \tl_clear:N \blx@trackkeys@foot
and all of them do:
1021         \cs_set_eq:NN \blx@lastmpfn \z@
1022     }

```

(End of definition for __postnotes_biblatex_citereset_local:.)

```

1023 }

```

biblatex’s `refsections`, contrary to `refsegments` and `refcontexts` which are handled in the L^AT_EX 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`.

```

1024 \AddToHook { package/biblatex/after }
1025 {
1026     \tl_new:N \l__postnotes_biblatex_orig_refsection_tl
1027     \tl_new:N \g__postnotes_biblatex_prev_refsection_tl
1028     \AddToHook { postnotes/print/begin } [ postnotes ]
1029     {
1030         \tl_set:Nx \l__postnotes_biblatex_orig_refsection_tl
1031         { \int_use:N \c@refsection }
1032         \tl_gset:Nx \g__postnotes_biblatex_prev_refsection_tl
1033         \l__postnotes_biblatex_orig_refsection_tl
1034     }
1035     \AddToHook { postnotes/print/note/begin } [ postnotes ]
1036     {
1037         \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1038         { biblatex@refsection } \l__postnotes_restore_tmp_tl
1039         \tl_if_eq:NNF
1040         \l__postnotes_restore_tmp_tl
1041         \g__postnotes_biblatex_prev_refsection_tl
1042         {

```

```

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

```

1057 \tl_new:N \l__postnotes_note_zlabel_tl

```

(End of definition for `\l__postnotes_note_zlabel_tl`.)

```

1058 \AddToHook { package/zref-user/after }
1059 {

```

Provide `zlabel` option.

```

1060     \keys_define:nn { postnotes/note }
1061     {
1062         zlabel .tl_set:N = \l__postnotes_note_zlabel_tl ,
1063         zlabel .value_required:n = true ,
1064     }

```

`\postnotezref` Provide `\postnotezref`.

```

\postnotezref{*}{\label}

```

```

1065 \NewDocumentCommand \postnotezref { s m }
1066 { \__postnotes_note_zref:nn {#1} {#2} }

```

(End of definition for `\postnotezref`.)

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

```

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

```

```

1067 \cs_new_protected:Npn \__postnotes_note_zref:nn #1#2
1068 {
1069     \group_begin:
1070     \__postnotes_typeset_mark_wrapper:n
1071     {
1072         \bool_lazy_all:nTF
1073         {

```

```

1074         { ! #1 }
1075         { \l__postnotes_hyperlink_bool }
1076         { \l__postnotes_zrefhyperref_bool }
1077     }
1078     {
1079         \hyperlink
1080         { \zref@extractdefault {#2} { anchor } { } }
1081         { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1082     }
1083     { \__postnotes_make_mark:nnn { \zref{#2} } { } { } }
1084 }
1085 \group_end:
1086 }

```

(End of definition for __postnotes_note_zref:nn.)

```

1087 }

1088 \bool_new:N \l__postnotes_zrefhyperref_bool
1089 \AddToHook { package/zref-hyperref/after }
1090 { \bool_set_true:N \l__postnotes_zrefhyperref_bool }

```

zref-clever

```

1091 \AddToHook { package/zref-clever/after }
1092 {
1093     \zcsetup
1094     {
1095         countertype = { postnote = endnote } ,
1096         countertype = { postnotetext = endnote } ,
1097     }
1098     \AddToHook { postnotes/print/begin } [ postnotes ]
1099     { \zcsetup { counterresetby = { postnotetext = postnotessection } } }
1100 }

```

zref-check

```

1101 \AddToHook { package/zref-check/after }
1102 {
1103     \IfPackageAtLeastTF { zref-check } { 2022-07-05 }
1104     {
1105         \AddToHook { postnotes/note/store } [ postnotes ]
1106         {
1107             \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1108             { zref-check@abschap } { \int_use:N \c@zc@abschap }
1109             \prop_gput:cnx { \__postnotes_data_name:e { \l__postnotes_note_id_tl } }
1110             { zref-check@abssec } { \int_use:N \c@zc@abssec }
1111         }
1112         \AddToHook { postnotes/print/note/begin } [ postnotes ]
1113         {
1114             \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1115             { zref-check@abschap } \l__postnotes_restore_tmp_tl
1116             \int_set:Nn \c@zc@abschap { \l__postnotes_restore_tmp_tl }
1117             \__postnotes_prop_get:nnN { \l__postnotes_print_note_id_tl }
1118             { zref-check@abssec } \l__postnotes_restore_tmp_tl
1119             \int_set:Nn \c@zc@abssec { \l__postnotes_restore_tmp_tl }

```

```

1120     }
1121   }
1122   { }
1123 }

```

amsmath

```

1124 \AddToHook { package/amsmath/after }
1125 {

```

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

```

1126   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1127   {
1128     \legacy_if:nT { measuring@ }
1129     {
1130       \bool_set_true:N \l__postnotes_inhibit_note_bool
1131       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1132       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1133     }
1134   }

```

However, the `\text` macro, defined by `amstext` (required by `amsmath`), poses problems if its own. Despite my best efforts, I could not salvage things from the use of `\mathchoice` and the redefinitions of `\setcounter` and `\addtocounter` performed by `amstext`. Setting `\l__postnotes_maybe_multi_bool` when `firstchoice@` is false grants us a working situation for display style. But the use of `\postnote` inside `\text` (and, if `amsmath` is loaded, `\textnormal`, `\textup`, etc.) in inline math environments is not supported. If a note really needs to be there, one can use the `nomark` option and `\postnoteref`. Things should work in text mode and in display style. For some related discussion with regard to footnotes, see <https://tex.stackexchange.com/a/82820> and, in particular, Barbara Beeton’s comment: “This is certainly bravura code. I do hope it doesn’t result in a request to add `\footnote` capabilities to `amsmath`’s multi-line display facilities. (The answer will almost certainly be no. We agree with Kopka & Daly.)”

```

1135   \AddToHook { postnotes/note/begin } [ postnotes ]
1136   {
1137     \legacy_if:nF { firstchoice@ }
1138     { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1139   }
1140 }

```

csquotes

```

1141 \AddToHook { package/csquotes/after }
1142 {
1143   \bool_new:N \l__postnotes_csquotes_measuring_bool
1144   \BlockquoteDisable
1145   { \bool_set_true:N \l__postnotes_csquotes_measuring_bool }
1146   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1147   {
1148     \bool_if:NT \l__postnotes_csquotes_measuring_bool
1149     {
1150       \bool_set_true:N \l__postnotes_inhibit_note_bool
1151       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1152       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1153     }

```

```

1154     }
1155 }

```

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>.

```

1156 \AddToHook { package/tabularx/after }
1157 {
1158   \bool_new:N \l__postnotes_tabularx_inside_env_bool
1159   \AddToHook { env/tabularx/begin } [ postnotes ]
1160   {
1161     \bool_set_true:N \l__postnotes_tabularx_inside_env_bool
1162     \cs_set_eq:NN \__postnotes_tabularx_saved_write:Nn \write
1163   }
1164   \AddToHook { postnotes/note/inhibit } [ postnotes ]
1165   {
1166     \bool_lazy_and:nnT
1167       { \l__postnotes_tabularx_inside_env_bool }
1168       { ! \cs_if_eq_p:NN \write \__postnotes_tabularx_saved_write:Nn }
1169     {
1170       \bool_set_true:N \l__postnotes_inhibit_note_bool
1171       \bool_set_true:N \l__postnotes_print_plain_mark_bool
1172       \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1173     }
1174   }
1175 }

```

tabularray

```

1176 \AddToHook { package/tabularray/after }
1177 {

```

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

```

1178   \bool_if_exist:NTF \lTblrMeasuringBool
1179   {

```

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

```

1180     \AddToHook { postnotes/note/inhibit } [ postnotes ]
1181     {
1182       \bool_if:NT \lTblrMeasuringBool

```

```

1183         {
1184             \bool_set_true:N \l__postnotes_inhibit_note_bool
1185             \bool_set_true:N \l__postnotes_print_plain_mark_bool
1186             \bool_set_true:N \l__postnotes_print_plain_mark_stepcounter_bool
1187         }
1188     }
1189 }
1190 {

```

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

```

1191     \clist_map_inline:nn
1192     {
1193         tblr , longtblr , talltblr , booktabs ,
1194         longtabs , talltabs , +array
1195     }
1196     {
1197         \AddToHook { env/#1/begin } [ postnotes ]
1198         { \bool_set_true:N \l__postnotes_maybe_multi_bool }
1199     }
1200 }
1201 }

```

10 Languages

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

`\pnhdnotes`

`\pnhdtopage`

`\pnhdtopages`

```

1202 \tl_new:N \pntitle
1203 \tl_new:N \pnhdnotes
1204 \tl_new:N \pnhdtopage
1205 \tl_new:N \pnhdtopages
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 }

```

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

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

```

\__postnotes_define_language:nn {⟨postnote language⟩} {⟨setup⟩}
1210 \cs_new_protected:Npn \__postnotes_define_language:nn #1#2
1211 {
1212     \tl_new:c { g__postnotes_language_ #1 _tl }
1213     \tl_gset:cn { g__postnotes_language_ #1 _tl } {#2}
1214 }

```

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

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

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

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

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

```

\__postnotes_set_babel_language:nn {\<babel language>} {\<postnote language>}

1215 \cs_new_protected:Npn \__postnotes_set_babel_language:nn #1#2
1216 {
1217   \ActivateGenericHook { babel/#1/beforeextras }
1218   \exp_args:Nnv \AddToHook { babel/#1/beforeextras }
1219   { g__postnotes_language_ #2 _tl }
1220 }

```

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

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

`__postnotes_set_polyglossia_language: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>}

1221 \cs_new_protected:Npn \__postnotes_set_polyglossia_language:nn #1#2
1222 {
1223   \AddToHook { package/polyglossia/after }
1224   {
1225     \exp_args:Nnv \csgappto { captions #1 }

```

```

1226         { g__postnotes_language_ #2 _tl }
1227     }
1228 }

```

(End of definition for __postnotes_set_polyglossia_language:nn.)

English

```

1229 \__postnotes_define_language:nn { english }
1230 {
1231     \tl_set:Nn \pntitle      { Notes }
1232     \tl_set:Nn \pnhdnotes   { Notes }
1233     \tl_set:Nn \pnhdtopage  { to~page }
1234     \tl_set:Nn \pnhdtopages { to~pages }
1235 }
1236 \__postnotes_set_babel_language:nn { english } { english }
1237 \__postnotes_set_babel_language:nn { british } { english }
1238 \__postnotes_set_babel_language:nn { american } { english }
1239 \__postnotes_set_babel_language:nn { canadian } { english }
1240 \__postnotes_set_babel_language:nn { australian } { english }
1241 \__postnotes_set_babel_language:nn { newzealand } { english }
1242 \__postnotes_set_babel_language:nn { UKenglish } { english }
1243 \__postnotes_set_babel_language:nn { USenglish } { english }
1244 \__postnotes_set_polyglossia_language:nn { english } { english }

```

Portuguese

```

1245 \__postnotes_define_language:nn { portuguese }
1246 {
1247     \tl_set:Nn \pntitle      { Notas }
1248     \tl_set:Nn \pnhdnotes   { Notas }
1249     \tl_set:Nn \pnhdtopage  { da~página }
1250     \tl_set:Nn \pnhdtopages { das~páginas }
1251 }
1252 \__postnotes_set_babel_language:nn { portuguese } { portuguese }
1253 \__postnotes_set_babel_language:nn { brazilian } { portuguese }
1254 \__postnotes_set_babel_language:nn { portuges } { portuguese }
1255 \__postnotes_set_babel_language:nn { brazil } { portuguese }
1256 \__postnotes_set_polyglossia_language:nn { portuguese } { portuguese }

```

French

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

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

```

1257 \__postnotes_define_language:nn { french }
1258 {
1259     \tl_set:Nn \pntitle      { Notes }
1260     \tl_set:Nn \pnhdnotes   { Notes }
1261     \tl_set:Nn \pnhdtopage  { de~la~page }
1262     \tl_set:Nn \pnhdtopages { des~pages }
1263 }
1264 \__postnotes_set_babel_language:nn { french } { french }
1265 \__postnotes_set_babel_language:nn { acadian } { french }
1266 \__postnotes_set_polyglossia_language:nn { french } { french }

```


German

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

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

```

1267 \__postnotes_define_language:nn { german }
1268 {
1269   \tl_set:Nn \pntitle      { Endnoten }
1270   \tl_set:Nn \pnhdnotes   { Endnoten }
1271   \tl_set:Nn \pnhdtopage  { zu-Seite }
1272   \tl_set:Nn \pnhdtopages { zu-Seiten }
1273 }
1274 \__postnotes_set_babel_language:nn { german }      { german }
1275 \__postnotes_set_babel_language:nn { ngerman }     { german }
1276 \__postnotes_set_babel_language:nn { austrian }   { german }
1277 \__postnotes_set_babel_language:nn { naustrian }   { german }
1278 \__postnotes_set_babel_language:nn { swissgerman } { german }
1279 \__postnotes_set_babel_language:nn { nswissgerman } { german }
1280 \__postnotes_set_polyglossia_language:nn { german } { german }
1281 </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		371, 372, 485, 909, 1088, 1143, 1158	
\ActivateGenericHook	1217	\bool_set_false:N	
\addto	38, 39	143, 222, 231, 232, 247, 376, 377, 378	
\addtocounter	36	\bool_set_true:N	
\AddToHook	240, 907, 932, 937, 939, 941, 949, 951, 963, 968, 1024, 1028, 1035, 1058, 1089, 1091, 1098, 1101, 1105, 1112, 1124, 1126, 1135, 1141, 1146, 1156, 1159, 1164, 1176, 1180, 1197, 1218, 1223	148, 221, 226, 227, 356, 935, 1090, 1130, 1131, 1132, 1138, 1145, 1150, 1151, 1152, 1161, 1170, 1171, 1172, 1184, 1185, 1186, 1198	
\AddToHookNext	644	\bool_to_str:N	
B		46	
\begin	587	\bool_until_do:nn	
\BlockquoteDisable	1144	538	
bool commands:		box commands:	
\bool_gset_false:N	645	\box_use:N	
\bool_gset_true:N	530	292	
\bool_if:NTF	30, 245, 334, 380, 384, 400, 408, 479, 528, 533, 586, 636, 668, 912, 1148, 1182	\box_wd:N	
\bool_if_exist:NTF	1178	291	
\bool_lazy_all:nTF	1072	\l_tmpa_box	
\bool_lazy_and:nnTF	446, 655, 704, 1166	290, 291, 292	
\bool_new:N	136, 213, 214, 215, 267, 344, 347, 348, 370,	C	
		\caption	10, 11, 30
		\chapter	117
		\citereset	32
		clist commands:	
		\clist_map_inline:Nn	
			1009, 1012, 1015, 1018
		\clist_map_inline:nn	1191
		\counterwithin	12

\g__postnotes_header_page_first_- prop	728, 757, 814, 860	\l__postnotes_post_printnote_tl	144, 203, 210, 620
\g__postnotes_header_page_last_- prop	728, 758, 801, 842, 864	\l__postnotes_post_textmark_tl	202, 208, 670, 673
\g__postnotes_header_prev_last_- chap_tl	728, 766, 874, 879, 881	\l__postnotes_pre_textmark_tl	201, 206, 670, 673
\g__postnotes_header_prev_last_- name_tl	728, 768, 896, 901, 903	\l__postnotes_prev_mark_chap_tl	728, 772, 792, 806, 829, 847
\g__postnotes_header_prev_last_- page_tl	728, 765, 863, 868, 870	\l__postnotes_prev_mark_name_tl	728, 774, 796, 812, 833, 853
\g__postnotes_header_prev_last_- sect_tl	728, 767, 885, 890, 892	\l__postnotes_prev_mark_page_tl	728, 771, 790, 803, 827, 844
\g__postnotes_header_sect_first_- prop	728, 761, 820, 882	\l__postnotes_prev_mark_sect_tl	728, 773, 794, 809, 831, 850
\g__postnotes_header_sect_last_- prop	728, 762, 807, 848, 886	\l__postnotes_prev_text_page_tl	26, 27, 728, 769, 786, 799, 802, 805, 808, 811, 834, 840, 843, 846, 849, 852
\g__postnotes_header_vars_next_- bool	29, 530, 645, 909, 912	\l__postnotes_print_as_list_bool	136, 143, 148, 533, 586, 636, 668
\l__postnotes_hyperlink_bool	213, 221, 226, 231, 247, 408, 448, 656, 1075	\l__postnotes_print_content_tl	501, 576, 577, 603, 619
\l__postnotes_hyperref_warn_bool	214, 222, 227, 232, 245	\l__postnotes_print_counter_tl	501, 600, 606
__postnotes_inhibit_note:	374	\l__postnotes_print_env_tl	135, 145, 149, 587, 637
__postnotes_inhibit_note:TF	22, 318, 370	\l__postnotes_print_format_tl	128, 131, 589
\l__postnotes_inhibit_note_bool	14, 370, 376, 400, 1130, 1150, 1170, 1184	\l__postnotes_print_mark_tl	501, 597, 608, 617
__postnotes_list_makelabel:n	162, 179, 189	\l__postnotes_print_note_id_- next_tl	501, 548, 553, 555, 569, 572, 624, 629, 631
__postnotes_make_mark:nnn	193, 398, 410, 414, 451, 453, 1081, 1083	__postnotes_print_notes:	18, 19, 22–24, 495, 516, 516
__postnotes_make_text_mark:nnn	197, 659, 663	\l__postnotes_print_plain_mark_- bool	14, 371, 377, 380, 1131, 1151, 1171, 1185
\l__postnotes_manual_sortnum_- bool	30, 347, 356	\l__postnotes_print_plain_mark_- stepcounter_bool	14, 37, 372, 378, 384, 1132, 1152, 1172, 1186
\l__postnotes_mark_tl	25, 321, 324, 330, 337, 343, 351, 382, 387, 394, 398	\g__postnotes_print_postnotes_- int	501, 519, 918, 922
\l__postnotes_maybe_multi_bool	23, 36, 46, 348, 935, 1138, 1198	\l__postnotes_print_type_curr_tl	501, 543, 544, 580, 641
\l__postnotes_nomark_bool	334, 344, 365	\l__postnotes_print_type_next_tl	501, 549, 556, 558, 625, 632, 634
__postnotes_note:nn	12, 15, 312, 313, 314	\l__postnotes_print_type_prev_tl	501, 525, 579, 584, 640
\g__postnotes_note_id_int	12, 307, 320, 475	__postnotes_prop_gclear:n	4, 64, 71, 647
\l__postnotes_note_label_tl	346, 367, 434, 435	__postnotes_prop_get:nnN	4, 64, 64, 542, 554, 562, 565, 568, 571, 574, 595, 598, 601, 630, 681, 698, 699, 702, 703, 708,
__postnotes_note_ref:nn	16, 440, 441, 441		
\l__postnotes_note_zlabel_tl	34, 436, 437, 1057, 1062		
__postnotes_note_zref:nn	34, 1066, 1067, 1067		

709, 791, 793, 795, 828, 830, 832, 970, 973, 976, 979, 1037, 1114, 1117	_postnotes_tabularx_saved_write:Nn 1162, 1168
_postnotes_prop_item:nn 4, 64, 69, 778, 819, 822, 825	_postnotes_text_mark_wrapper:n 22, 609, 653, 666
\g_postnotes_queue_seq 12, 23, 25, 307, 326, 476, 520, 526, 527, 529, 531, 538, 540, 546, 552, 622, 628	_postnotes_typeset_mark:nn 15, 336, 404, 404, 417
\c_postnotes_ref_prefix_tl 4, 73, 75, 96, 97, 103, 104, 685	_postnotes_typeset_mark_wrapper:n 15, 397, 404, 406, 419, 444, 1070
\l_postnotes_restore_tmp_tl 931, 971, 972, 974, 975, 977, 978, 980, 981, 1038, 1040, 1044, 1046, 1115, 1116, 1118, 1119	_postnotes_typeset_text_mark:nn 22, 615, 653, 653, 665
\l_postnotes_saved_spacefactor_tl 418, 424, 429	_postnotes_verify_multipass:N 22, 23, 38, 527, 675, 675
\g_postnotes_sectid_int 44, 470, 474	\l_postnotes_zrefhyperref_bool 1076, 1088, 1090
_postnotes_section:nn 17, 458, 470, 471	\postnotessection 3, 16, 17, 457
\l_postnotes_section_exp_bool 479, 485, 490	\postnotessectionx 16, 457
\g_postnotes_section_name_tl 42, 477, 484, 488	\postnotesetup 9, 304
_postnotes_set_babel_language:nn 39, 1215, 1215, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1252, 1253, 1254, 1255, 1264, 1265, 1274, 1275, 1276, 1277, 1278, 1279	\postnotetext 11, 12
_postnotes_set_headers_vars:n 25, 28, 29, 857, 857, 906, 913, 919	\postnotezref 34, 1065
_postnotes_set_headers_vars_first: 24, 29, 532, 907, 915	prg commands:
_postnotes_set_headers_vars_next: 24, 29, 907, 908, 910	\prg_new_protected_conditional:Npnn 374
_postnotes_set_mark_page_label:n 4, 24, 76, 76, 81, 332	\prg_return_false: 402
_postnotes_set_polyglossia_language:nn 39, 1221, 1221, 1244, 1256, 1266, 1280	\prg_return_true: 401
_postnotes_set_print_page_label:n 4, 24, 76, 88, 93, 917	\printpostnotes 11, 12, 17–19, 22, 24, 26–29, 31, 32, 494
_postnotes_set_text_page_label:n 4, 24, 76, 82, 87, 613	prop commands:
_postnotes_set_user_labels: 34, 333, 432, 432	\prop_gclear:N 72, 757, 758, 759, 760, 761, 762, 763, 764
\l_postnotes_sort_bool 267, 270, 528	\prop_get:NnNTF 66, 860, 864, 871, 875, 882, 886, 893, 897
\l_postnotes_sort_num_fp 31, 345, 355	\prop_gput:Nnn 23, 24, 26, 28, 36, 39, 41, 43, 45, 47, 53, 56, 59, 61, 801, 804, 807, 810, 814, 817, 820, 823, 842, 845, 848, 851, 953, 955, 957, 960, 1107, 1109
_postnotes_sort_queue:N 23, 529, 693, 693	\prop_item:Nn 70
_postnotes_store:nn . 2, 19, 20, 339	\prop_new:N 22, 52, 728, 729, 730, 731, 732, 733, 734, 735
_postnotes_store_section:nn 3, 50, 50, 63, 480, 481	\providecommand 3
\l_postnotes_tabularx_inside_env_bool 1158, 1161, 1167	\ProvidesExplPackage 14

R

\ref 2, 451, 453
\refsection 33, 1051
\refstepcounter 12
\rightmargin 163, 180
\rightskip 283

S

scan commands:
\scan_stop: 430
\section 124

seq commands:			
\seq_clear:N	678	\blx@refcontext@context	961
\seq_get_left:NN	552, 628	\blx@refcontext@labelalphanametemplatenam	991, 998
\seq_gpop_left:NN	540	\blx@refcontext@labelprefix	986
\seq_gput_right:Nn	326, 476	\blx@refcontext@labelprefix@real	987
\seq_gset_eq:NN	690	\blx@refcontext@sortingnamekeytemplatenam	989, 995
\seq_gsort:Nn	696	\blx@refcontext@sortingtemplatenam	988, 994
\seq_if_empty:NTF	520, 546, 622	\blx@refcontext@uniquenametemplatenam	990, 997
\seq_if_empty_p:N	538	\blx@sorting	988
\seq_map_inline:Nn	646, 679, 775	\blx@theendnote	943
\seq_new:N	310, 510	\blx@theendnotetext	944
\seq_put_right:Nn	686, 688	\blx@trackhash@foot	1012, 1014
\seq_set_eq:NN	526	\blx@trackhash@text	1009, 1011
\l_tmpa_seq	678, 686, 688, 690	\blx@trackkeys@foot	1018, 1020
\setcounter	36, 515	\blx@trackkeys@text	1015, 1017
\setlength	159, 160, 161, 163, 164, 165, 166, 167, 168, 176, 177, 178, 180, 181, 182, 183, 184, 185, 283, 284, 285	\c@blx@maxsection	1043
\setlocalecaption	39	\c@page	85, 91, 913
skip commands:		\c@postnote	14, 27, 32, 391
\skip_horizontal:n	291	\c@postnotetext	605
\small	132	\c@refsection	954, 972, 1003, 1004, 1031
sort commands:		\c@refsegment	956, 975
\sort_return_same:	712, 714, 716	\c@zc@abschap	1108, 1116
\sort_return_swapped:	711	\c@zc@abssec	1110, 1119
\spacefactor	424, 429	\hyper@linkend	412, 661
\stepcounter	12, 323, 386, 560	\hyper@linkstart	411, 660
str commands:		\ifmeasuring@	36
\str_if_eq_p:nn	705, 706	\p@postnote	330, 608
		\post@note	4, 73, 79, 85, 91
		\z@	1021
		\zref@extractdefault	1080
T		\text	10, 11, 36
T _E X and L ^A T _E X 2 _ε commands:		\textnormal	36
\@afterindentfalse	535	\textup	36
\@afterindenttrue	19, 535, 536	\the	424
\@auxout	78, 84, 90	\thechapter	24, 37, 57
\@capttype	934	\theHpostnote	12
\@currentcounter	329, 604	\thepage	24, 79
\@currentlabel	330, 607	\thepostnote	14, 324, 387, 394
\@footnotemark	15	\thesection	24, 40, 60
\@ifl@t@r	3	tl commands:	
\@makefnmark	7	\c_empty_tl	105
\@mkboth	118, 125	\tl_clear:N	67, 98, 769, 770, 771, 772, 773, 774, 986, 987, 1003, 1004, 1011, 1014, 1017, 1020
\@newl@bel	4, 75	\tl_const:Nn	73
\@textsuperscript	196, 290	\tl_gclear:N	477, 749, 750, 751, 752, 753, 754, 755, 756, 765, 766, 767, 768
\blx@bibfiles	33	\tl_gput_right:Nn	39
\blx@edef@refcontext	981, 992	\tl_gset:Nn	862, 863, 867, 868, 870, 873, 874, 878,
\blx@endrefsection	1050		
\blx@err@endnote	945		
\blx@info	1049		
\blx@lasthash@foot	1008		
\blx@lasthash@text	1007		
\blx@lastkey@foot	1006		
\blx@lastkey@text	1005		
\blx@lastmpfn	1021		

879, 881, 884, 885, 889, 890, 892, 895, 896, 900, 901, 903, 1032, 1213	681, 682, 698, 700, 702, 705, 708, 710, 861, 862, 865, 867, 868, 872, 873, 876, 878, 879, 883, 884, 887, 889, 890, 894, 895, 898, 900, 901
\tl_gset_eq:NN 1045	\l_tmpb_tl . 699, 700, 703, 706, 709, 710
\tl_if_empty:NTF 321, 382, 434, 436, 783, 799, 840	\togglefalse 985
\tl_if_eq:NNTF ... 700, 785, 927, 1039	\toggletrue 940
\tl_if_eq:NnTF 544, 558, 584, 634, 682	token commands:
\tl_if_eq:nnTF 141, 777	\token_to_str:N 79, 85, 91
\tl_new:N 128, 135, 201, 202, 203, 308, 343, 346, 418, 484, 496, 497, 498, 499, 500, 502, 503, 504, 505, 506, 507, 508, 509, 720, 721, 722, 723, 724, 725, 726, 727, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 931, 1026, 1027, 1057, 1202, 1203, 1204, 1205, 1212	\topsep 167, 184
\tl_set:Nn 97, 144, 145, 149, 309, 324, 387, 394, 424, 525, 548, 549, 579, 624, 625, 640, 834, 988, 989, 990, 991, 1030, 1206, 1207, 1208, 1209, 1231, 1232, 1233, 1234, 1247, 1248, 1249, 1250, 1259, 1260, 1261, 1262, 1269, 1270, 1271, 1272	U
\l_tmpa_tl 681, 682, 698, 700, 702, 705, 708, 710, 861, 862, 865, 867, 868, 872, 873, 876, 878, 879, 883, 884, 887, 889, 890, 894, 895, 898, 900, 901	\undef 1005, 1006, 1007, 1008
	use commands:
	\use:N 978
	\use_none:n 1048, 1049
	\UseHook 48, 328, 379, 524, 592
	W
	\write 1162, 1168
	Z
	\zcsetup 1093, 1099
	\zlabel 437
	\zref 1081, 1083